Test, Evaluierung und Auswahl eines Netzwerkdateisystems

nach den Anforderungen der Hochschule Mittweida

Diplomarbeit Hochschule Mittweida

Fachbereich Informatik

vorgelegt von:	Ronny Gubatz
Fachbereich:	Informatik
Matrikelnummer:	14312
Erstgutachter:	Prof. Dr.-Ing. Joachim Geiler
Zweitgutachter:	Dipl.-Inf. Matthias Lühr

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

A

ACL Access Controll List, Zugriffsschutzliste um den Zugriff auf Daten Anhand Berechtigungen zu regeln

Apple Firma aus den USA die Computer und Unterhaltungselektronik herstellt

B

Beta Status Entwicklungsstadium in der Softwareentwicklung der nicht für den produktiven Einsatz gedacht ist

Broadcast Ein Netzwerkrundruf an alle Teilnehmer in einem lokalen Netzwerk

D

Datei ist ein strukturierter, inhaltlich zusammen gehörender Bestand von Daten

Dateiname identifiziert eine Datei bei Datenübertragung oder auf einem Datenträger. Dateinamen sind meist in ihrer Länge beschränkt

DNS Domain Name System, ein TCP/IP Dienst, der die Namensauflösung realisiert.

E

ext2 (second extended filesystem) war viele Jahre lang das Standarddateisystem des Linux-Betriebssystems

F

Festplattensubsysteme Gehäuse oder Plattform, in denen Festplatten eingebaut werden. Mittels RAID Technologie wird ein Ausfall einzelner Festplatten kompensiert. Das System wird mit Fiber Channel oder TCP/IP angebunden.

FFS Das Berkeley Fast File System ist ein Dateisystem, welches vom Original-Dateisystem FS abstammt, das ursprünglich von AT&T für die ersten Versionen von UNIX (System V) entwickelt wurde.

H

HPC High Performance Computing

I

IBM Public License Eine Open Source Lizenz aus dem Hause IBM, die im August 1999 von der Open Source Initiative (OSI) als Open Source Lizenz akzeptiert wurde

K

Kernelmode Programmcode der mit der höchste Privilegierungsstufe läuft

Kohärenzprotokoll Ein Kohärenzprotokoll hat die Aufgabe, den Status eines Speicherblocks zu verfolgen und zu korrigieren um mehrere Versionen eines Datenbestandes zu verhindern.

M

MacOSX proprietäre Distribution des frei erhältlichen Darwin-Betriebssystems der Firma Apple

Man-in-the-Middle Eine Angriffsform in Netzwerken, bei welcher sich der Angreife gegenüber dem Angegriffenen als vertrauenswürdige Instanz ausgibt.

Metadaten bezeichnet Daten, die Daten über andere Daten enthalten. Beispiel: Author und Verlag eines Buches

MNTV3 Mount Protokoll Version 3, authentifiziert Benutzer am Server. Im RFC1813 spezifiziert.

N

NAS Network attached Storage

Needham-Schroeder-Protokoll Protokoll zum sicheren Datenaustausch in einem dezentralen Netzwerk. Es vereint Schlüsselaustausch und Authentifikation mit dem Ziel, eine sichere Kommunikation zwischen zwei Parteien in einem dezentralen Netzwerk zu etablieren.

NetBIOS over TCP/IP Auch **NetBT** oder **NBT** genannt. Das Netzwerkprotokoll NetBIOS wird über TCP/IP angesprochen.

NFS Network File System der Firma Sun Microsystems

NLM Network Locking Manager, Implementiert die `lockf()` Funktion unter NFS

O

OSI Schichtenmodell Open Systems Interconnection Reference Model wird ein Schichtenmodell der Internationalen Standardisierungsorganisation (ISO) bezeichnet

P

POSIX Portable Operating System Interface

R

RPC (remote procedure call) Protokoll um Systemaufrufe über das Netzwerk zu realisieren

RPC2 RPC2 ist ein portables und erweitertes RPC Paket aufbauend auf IP/UDP.

RX Netzwerkprotokoll um über unstabile MoDem-Verbindungen große Datenmengen zu übertragen

S

SSH (Secure Shell) eine Software, um verschlüsselte und authentifizierte Netzwerkverbindungen zu schaffen.

T

TCP/IP Transmission Control Protocol / Internet Protocol

U

UDP (user datagram protocol) ist ein minimales, verbindungsloses Netzwerkprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört

UNIX ist ein Mehrbenutzer-Betriebssystem. Es wurde Anfang der 1970er Jahre von Bell Laboratories entwickelt

URI Uniform Resource Identifier engl. einheitlicher Bezeichner für Ressourcen

1 Vorwort

1.1 Voraussetzungen zum Verständnis der Arbeit

Um die nachfolgende Arbeit zu verstehen, sollte der Leser wissen wie das OSI Schichtenmodell aufgebaut ist und mit den Grundlagen von IP Netzwerken vertraut sein.

1.2 Typographische Konventionen

In der folgenden Tabelle werden die Bedeutungen der in dieser Arbeit verwendeten typografischen Auszeichnungen erklärt.

Schriftart oder Schriftstil	Bedeutung	Beispiel
Italic	Begriffe die im Glossar erklärt werden	*POSIX*
`Monospace`	Befehle und Datei- und Verzeichnisnamen	`ls -a`
Fettschrift	Benutzereingaben	systemname% **su**

Abbildung 1.1: typographische Konventionen

2 Aufgabenstellung

2.1 Derzeitige Situation

Alle Studenten und einige Mitarbeiter der Hochschule Mittweida, bekommen die Möglichkeit, an Rechnern mit dem Betriebssystem Linux zu arbeiten.

Jeder Student bekommt seinen eigenen Speicherplatz nachfolgend Homebereich genannt, zugewiesen auf dem er seine Daten speichern kann.

Für die Windows Rechner wird der Homebereich von einem Novell Server über das Netzwerk exportiert. Dieser gibt ein Verzeichnis je Student frei und bei einer erfolgreichen Anmeldung wird das Verzeichnis vom Server fest in die Ordnerstruktur der Windows Arbeitsstation eingebunden.

Dadurch wird sichergestellt, daß der Anwender an jedem Rechner Zugriff auf seine persönlichen Daten hat.

Für die Linux Rechner wird der einheitliche Homebereich derzeitig mit dem Netzwerkdateisystem *NFSv3* realisiert. Nach einigen Jahren Betrieb haben sich einige gravierende Nachteile von NFSv3 herausgestellt und es muss ein neues Netzwerkdateisystem gefunden werden.

Die Server werden in einem Cluster betrieben. Sollte ein Server ausfallen, bleiben die Daten erreichbar. Über ein *SAN* greifen die Server auf die eigentlichen Daten zu.

Da es sich bei dem Server um eine virtuelle Maschine handelt, wird der Betrieb und die Umschaltung des Clusters von dem Wirtbetriebsystem verwaltet und ist nicht Bestandteil dieser Arbeit.

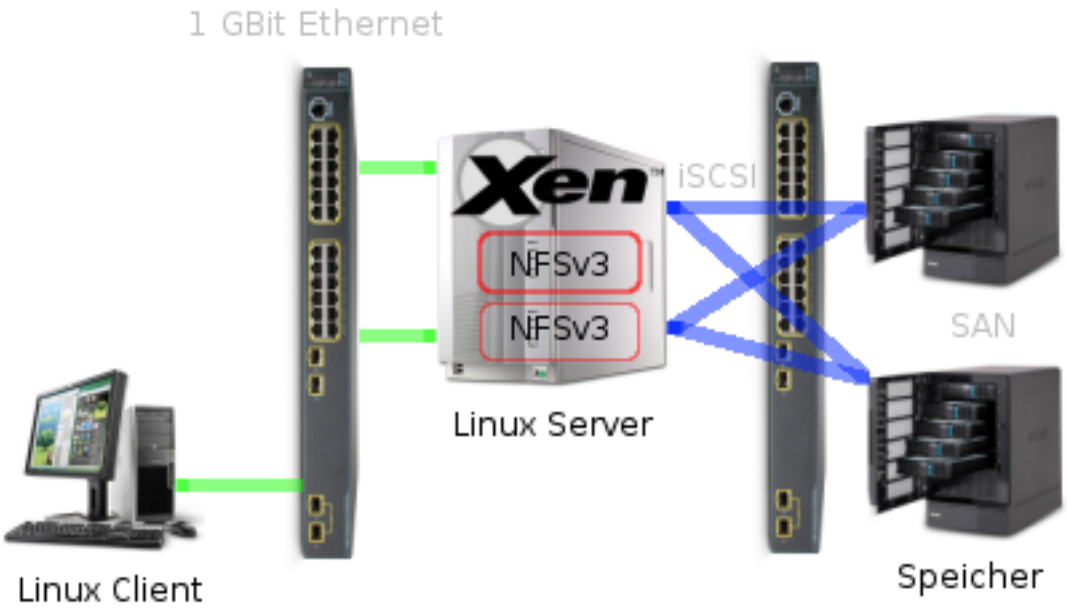

Abbildung 2.1: NFS Server an der HTWM

2.2 Nachteile des derzeitigen Netzwerkdateisystems NFSv3

An allen Linux Arbeitsstationen sind zu jeder Zeit die kompletten, exportierten Verzeichnisse des Server eingebunden. Das bedeutet für den NFS Server eine höhere Last.

Der Zugriff auf die Homebereiche aller Nutzer wird nur durch die lokale Sicherheit auf dem Client beschränkt. Sollte ein Anwender in der Lage sein, sich auf einem Linux System Root Rechte zu geben, gibt es keine Zugriffskontrolle mehr.

Ein Datenspion kann mit entsprechendem physikalischen Zugang und seinem eigenen Rechner die exportierten Verzeichnisse mounten und lesend wie auch schreiben auf alle Homebereiche zugreifen.

Der Datenspion kann sich auch als Server ausgeben um den Clienten einen Homebereich anzubieten, welcher unter Kontrolle des Angreifers steht.

Die Verwaltung der Firewallregeln für NFSv3 ist dadurch erschwert, da der **portmapper** Ports dynamisch festlegt.

2.3 Anforderungen an ein neues Netzwerkdateisystem

Das neue Netzwerkdateisystem soll nach dem Client-Server Prinzip arbeiten. Es soll für die Betriebssysteme Windows, Linux und MacOSX ein Client verfügbar sein. Um Identitätsmissbrauch zu vermeiden, muss das Dateisystem eine sichere und strenge Nutzerauthentifizierung einhalten. Die neue Lösung muss einen geeigneten Schutz gegen *Man in the Middle* Attacken aufweisen. Das heißt, über ein dokumentiertes kryptografisches Verfahren wird sichergestellt, daß sich zu keiner Phase der Kommunikation ein Angreifer als Kommunikationspartner ausgeben kann.

Es soll der Homebereich eines Benutzers erst nach der erfolgreichen Anmeldung am Rechner einbunden werden.

Auf dem Client Rechner gibt es keinen Unterschied ob eine Datei lokal gespeichert wird oder auf dem angebotenen Netzlaufwerk. Das Netzwerkdateisystem erfüllt den POSIX Standart.

Die Benutzer an der Hochschule Mittweida werden mittels einer zentralen Software auf Basis von *LDAP* und *PostGreSQL* verwaltet. Die neue Software muss sich nahtlos in diese Lösung integrieren lassen.

Zur Arbeit gehört die Implementierung eines Prototypen auf Basis einer *XEN VM* mit dem Betriebssystem Debian Lenny.

2.4 Ziel der Arbeit

Nach Fertigstellung der Arbeit soll an der Hochschule Mittweida ein geeignetes Netzwerkdateisystem aus einer Vielzahl gefunden und implementiert sein.

Durch die beschriebene Diskussion ist der Entscheidungsweg transparent und unter Berücksichtigung der zu diesem Zeitpunkt verfügbaren Technik, immer nachvollziehbar.

3 Theoretischer Hintergrund

[Rechenberg 1999] [Tannenbaum 1994]

In modernen Computersystemen ist es notwendig, dass Informationen persistent und zuverlässig gespeichert werden. Der Anwender soll eine abstrakte und vereinfachte Sicht auf seine Daten erhalten und logische Operationen wie lesen, bearbeiten oder löschen, ohne Wissen der genauen Hintergründe und Organisation des Datenträgers durchführen können. Das Computersystem muss sicherstellen, daß die Informationen

- in großen Mengen abgelegt und wiedergefunden
- gleichzeitig von mehreren Prozessen gelesen
- effizient auf dem Datenträger gespeichert

werden können.

Um die oben genannten Anforderungen zu erfüllen, werden Daten in kleinen Sektionen getrennt nach Typ und Inhalt in *Dateien* gespeichert. Somit kann der Anwender mit verschiedenen Prozessen und Programmen, auf seine Daten zugreifen. Der Anwender benutzt Dateien über ihren *Dateinamen*. Das Computersystem muss sicherstellen, daß nur durch einen expliziten Löschbefehl Dateien gelöscht werden. Die Organisation der Dateiarbeit wird in modernen Computersystemen durch das Betriebssystem realisiert. So schreibt Tannenbaum [1994]:

> „Dateien werden durch das Betriebssystem verwaltet. Wie sie strukturiert, benannt, zugegriffen, benutzt, geschützt und implementiert werden, sind wesentliche Punkte beim Entwurf von Betriebssystemen. Als Ganzes betrachtet ist dieser Teil des Betriebssystems, der sich mit Dateien beschäftigt, als **Dateisystem** bekannt. “

3.1 Dateisystem

Das Dateisystem ist ein elementarer Bestandteil des Betriebssystems um Informationen auf Datenträgern zu speichern. Zum aktuellen Zeitpunkt (Stand: März 2010) haben sich Festplatten in Form von rotierenden, ferromagnetischen Scheiben auf denen Informationen mittels Magnetisierung der Oberfläche dauerhaft gespeichert und abgerufen werden, als Datenträger etabliert.

Bevor Daten auf die Festplatte geschrieben werden, kodiert der Festplattencontroller die Daten, um eine höhere Speicherdichte auf dem Medium zu erreichen.

Die Daten werden in 512 Byte große Blöcke geteilt und nicht linear in Form einer Spur auf die Magnetscheibe geschrieben. Die Blockgröße von 512 Byte ist die kleinste Einheit, die vom Speichermedium gelesen oder geschrieben werden kann.

Beispielsweise wird eine Datei mit der Größe von 5120 Byte in 10 Blöcke, die scheinbar wahllos auf der Festplatte verteilt sind, gespeichert. Die scheinbar wahllose Anordnung wird vom Festplattencontroller dahingegen optimiert, daß der Festplattenkopf beim lesen der Datei in seiner Bewegung den kürzesten Weg zurücklegt.

Mehr dazu ab Kapitel 4.3.1 in Tannenbaum [1994].

3.1.1 Anforderungen an ein modernes Dateisystem

Die Anforderungen an ein Dateisystem sind abhängig vom Anwendungsfall. Beispielsweise sind die Anforderungen auf einem Supercomputer, der viele parallele Zugriffe auf verschiedenste Bereiche des Dateisystems bewerkstelligen muss, andere als die eines Dateisystems auf einem eingebetteten Computersystem ohne Benutzerschnittstelle, welches auch mit Dateinamen die maximal 32 Zeichen lang sind, auskommt.

Auf einem heutigen durchschnittlichen Anwendercomputer sollte ein modernes Dateisystem folgende Eigenschaften erfüllen:

- Datumsinformationen wie Erstellungsdatum, letzte Änderung und letzter Zugriff
- Groß- / Kleinschreibung und Sonderzeichen für Dateinamen ermöglichen
- Rechteverwaltung zur Zugriffssteuerung auf Dateien/Verzeichnisse

Um diese Anforderungen zu erfüllen, muss das Dateisystem diese Daten über Dateien und Verzeichnisse, die sogenannten *Metadaten* speichern. Die Verwaltung der Metadaten ist eine elementare Aufgabe eines Dateisystems.

Mehr dazu in Kapitel 4.1.5 in Tannenbaum [1994].

3.1.2 Grenzen moderner Dateisysteme

Die Anforderungen bilden die Grundlage für den Entwurf und das Design eines modernen Dateisystems. Das Format legt fest, wie und welche Metadaten das Dateisystem vorhält. Dadurch entstehen wiederum Beschränkungen, da der Speicherplatz für die Metadaten endlich ist.

Die Zuweisung von Metadaten wird bei herkömmlichen Dateisystemen bei der Erstellung vorgenommen. Dadurch werden die Grenzen zu diesem Zeitpunkt schon festgelegt.

Eine Ausnahme bildet hier das ZFS von Sun [2009]:

> Da ZFS Metadaten nach Bedarf zuweist, muss vorher kein zusätzlicher Speicherplatz reserviert werden, und die Dateianzahl wird lediglich durch den verfügbaren Speicherplatz begrenzt.

Dateisysteme unterscheiden sich durch ihre Grenzen:

- maximale Größe einer Datei
- maximale Länge eines Dateinamens
- maximale Größe des gesamten adressierbaren Speichers

Nachfolgende Tabellen stellen einige verbreitete Dateisysteme und ihrer Grenzen gegenüber.

Name	ext2	FFS	ext4
max. Dateigröße	2 TiB	4 GiB	16 TiB
max. Dateinamenlänge	255 Byte	255 Byte	256 Byte
max. Gesamtspeichergröße	32 TiB	256 TiB	1 EiB (2^{60}bytes)

Tabelle 3.1: Kennwerte gängiger Dateisysteme

Name	FAT16	FAT32	NTFS
max. Dateigröße	2 GiB	4 GiB	16 EiB (2^{60}bytes)
max. Dateinamenlänge	255 Zeichen	255 Zeichen	255 Zeichen
max. Gesamtspeichergröße	2 GiB	8 TiB (2^{40}bytes)	16 EiB (2^{60}bytes)

Tabelle 3.2: Kennwerte gängiger Dateisysteme der Firma Microsoft

3.2 Netzwerkdateisystem

In einer großen Umgebung wie der Hochschule Mittweida, wird es immer wieder geschehen, daß Anwender gezwungen sind, ihren Arbeitsplatz zu wechseln. Dabei kann nicht vom Anwender erwartet werden, daß er seine Daten in irgendeiner Form physisch von einem zum nächsten Ort transportiert. Die Daten werden an einer zentralen Stelle abgelegt und über einen standardisierten Mechanismus kann der Anwender auf diese zugreifen. Idealer Weise greift der Anwender auf die Daten genauso wie auf lokale Daten zu, somit ist das Netzwerkdateisystem transparent.

Eine zentrale Datenhaltung hat zudem folgende Vorteile:

- vereinfachte Datensicherung
- vereinfachte Suche nach Trojanern und Viren in den Nutzerdaten
- Möglichkeit viel Speicherplatz zur Verfügung zu stellen
- einfacher Austausch einer Arbeitsstation

Ein Nachteil der zentralen Datenhaltung ist die Abhängigkeit an ein funktionierendes Netzwerk und eines funktionierenden Servers. Durch einen redundanten

Aufbau der Netz- und Serverinfrastruktur wird in der Praxis die Ausfallzeit deutlich verringert.

Durch die zentrale Datenhaltung enstehen wieder neue Probleme, die von der eingesetzten Serversoftware gelöst werden müssen. Das sind im Wesentlichen:

3.2.1 Kohärenzproblem

Um die Netzwerklatenz zu überbrücken, werden Zwischenspeicher oder auch Puffer eingesetzt. Abhängig vom verwendeten Netzwerkdateisystem, kann dieser Puffer auf dem Server oder Client, oder auch auf beiden Seiten, implementiert sein. Durch verzögertes Schreiben können Inkonsistenzen zwischen den physikalisch gespeicherten Daten und denjenigen im Puffer enstehen. Das Netzwerkdateisystem muss mittels eines geeigneten *Kohärenzprotokolls* sicherstellen, daß alle Dateien in einem konsistenten Zustand gehalten werden.

Es gibt die folgenden Lösungsansätze:

- **Write-Through** Jeder Schreibzugriff auf eine Datei im Puffer wird sofort an den Server übertragen. Nachteilig ist, daß jegliche Effizienzsteigerung durch das Caching beim Schreibzugriff verloren geht.
- **Delayed-Write** Mehrere Änderungen werden zusammengefasst an den Server übertragen. Dadurch werden Netzzugriffe gespart, aber es entstehen auch größere Zeitschlitze, in denen durch Fehler inkonsistente Zustände entstehen.
- **Write-on-Close** Die Datei wird erst mit dem Server abgeglichen, nachdem sie geschlossen wurde.

So schreibt Schneider [2005]:

> Eine sichere Strategie in Bezug auf Benachrichtigung aller Server und Clients wäre eine zentrale Koordinierung, die alle Cacheeinträge aller Clients kennt. Diese Koordinierungsstelle würde von jedem Schreibzugriff in Kenntnis gesetzt und daraufhin alle Clients informieren, die eine betroffene Datei im Cache halten. Diese Art des Koheränz-Algorithmus ist jedoch weder robust gegen Ausfälle, noch gut skalierbar.

Somit besteht die optimale Lösung in einem Kompromiss aus den oben genannten.

3.2.2 Zugriffsschutz

In einem verteilten System muss der Server sicherstellen, daß

- die Identität des Clienten zuverlässig ermittelt wird
- die Zugriffsberechtigung für die Ressource vorhanden ist

Die Implementierung des Zugriffsschutzes ist ein wesentliches Merkmal in einem verteilten System.

So schreibt Rechenberg [1999] auf Seite 234:

> Es gibt prinzipiell drei Methoden, um die Identität zu beweisen: durch *Besitz*, z.B. einer Chipkarte, durch *Wissen*, z.B. eines Kennwortes, oder durch *Merkmale*, z.B. einen Fingerabdruck.

Im Wesentlichen werden in verteilten Systemen die Kommunikationspartner mit einer Kombination aus Benutzername und Kennwort authentifiziert. So auch die bisherige Praxis an der Hochschule Mittweida. Jeder Student verwendet seine Kombination aus Benutzername und Passwort. Letzteres muss alle 120 Tage geändert werden.

Die Übertragung darf natürlich nicht im Klartext über das Netzwerk erfolgen.

4 Gegenüberstellung verschiedener Netzwerkdateisysteme und Anforderungen

Die nachstehende Liste ist die Zusammenstellung von aktuellen Netzwerkdateisystemen der Wikipedia [2009] vom 25. November 2009.

Alle einzelnen Netzwerkdateisysteme werden in der Tabelle entsprechend der Anforderungen bewertet.

Ein + bedeutet, das Dateisystem erfüllt die Anforderung, bei einem - wird die Anforderung nicht erfüllt. Dateisysteme die mindestens 7 der geforderten 8 Anforderungen erfüllen, werden im der Arbeit praktisch Implementiert.

Das sind:

- OpenAFS
- CIFS/Samba
- Coda
- NFSv3
- NFSv4

Netzwerkdateisystem	klassisches Netzwerkdateisystem	Ortsunabhängig	Skalierbar	sichere Authentifizierung	Windows/Linux/Mac Client	Open Source	POSIX Semantik	Produktionsreif
OpenAFS	+	+	+	+	+	+	+	+
Apple Filing Protocol	+	-	-	+	-	-	+	+
CIFS/SMB	+	+	+	+	+	+	+	+
Coda	+	+	+	+	-	+	+	+
CFS	-	-	-	+	-	+	+	-
DCE/DFS	-	+	+	+	-	-	+	+
CXFS	-	-	+	+	-	-	+	+
DFS	-	+	+	+	+	-	+	+
EMC Celerra HighRoad	-	-	+	-	-	-	+	+
FhGFS	-	-	+	-	-	-	+	+
GFS	-	-	+	-	-	+	+	+
GlusterFS	-	-	+	-	-	+	+	+
GPFS	-	-	+	-	-	-	+	+
HAMMER	-	-	+	+	-	+	+	-
HP CFS	-	-	+	-	-	-	+	+
Lustre	-	-	+	-	-	+	+	+
Melio FS	-	-	+	-	-	-	+	+
Nasan File System	-	-	+	-	-	-	+	+
NFSv3	+	+	+	-	+	+	+	+
NFSv4	+	+	+	+	-	+	+	+
OCFS2	-	-	+	-	-	+	+	+
PSFS	-	-	+	-	-	-	+	+
PVFS2	-	-	+	+	-	+	+	-
QFS	-	-	+	+	-	+	+	-
StorNext File System	-	-	+	-	-	-	+	-
Veritas Storage Foundation Cluster File System	-	-	+	-	-	-	+	+
xFS	-	-	+	+	-	+	+	-
Xsan	-	-	+	-	-	-	+	+
XtreemFS	-	-	+	+	-	+	+	+

Tabelle 4.1: Gegenüberstellung Netzwerkdateisysteme / Anforderungen

5 Technische Betrachtung der ausgewählten Netzwerkdateisysteme

5.1 OpenAFS

AFS ist die Abkürzung für Andrew Filesystem.

5.1.1 Herkunft

AFS wurde an der Carnegie Mellon University (USA) im Jahr 1983 entwickelt. Der Gründer der Universität, Andrew Carnegie wurde mit dem Namen des Forschungsprojektes gewürdigt. Ab 1989 wurde das Dateisystem von der Firma Transarc Corporation kommerziell weiterentwickelt, bis 1998 IBM die Firma aufkaufte und ab 2000 den Quellcode von AFS unter der *IBM Public License* veröffentlichte. Das Andrew Dateisystem ist heute als offene Software unter dem Namen OpenAFS in der Version 1.4.11 (Stand November 2009) frei verfügbar.

5.1.2 Funktionsweise

Als Grundlagen bedienten sich die Entwickler eines Protokolls mit dem Namen *RX*, daß entwickelt wurde, um über unstabile Modem-Verbindungen große Datenmengen zu übertragen. Dabei wird das verbindungslose Protokoll *UDP* als Grundlage verwendet. Die Nachrichten werden von der Applikation gezählt und bei einem Übertragungsverlust nochmal per *RPC* angefordert. Mehr dazu von Nickolai Zeldovich in den Spezifikationen.

AFS funktioniert nach dem Client/Server-Modell, bei dem ein oder mehrere Server Daten anbieten und Clienten auf diese über ein Netzwerk zugreifen.

Unter AFS werden die Daten unter einem globalen Namensraum angeboten und sind somit unabhängig vom physikalischen Standort, Plattform oder Betriebssystem des Servers, immer unter der gleichen URL erreichbar.

Eine URL an der Hochschule Mittweida könnte z.B. so aussehen:

`/afs/minic.htwm.de/mni/benutzername`

Dabei wird in der AFS-Welt der Ordner `mni` als Zelle bezeichnet. Eine Zelle bildet die Menge der verwalteten Server einer Administration und stellt ein einziges großes Dateisystem dar. Mit Zellen könnten z.B. die einzelnen Fachbereiche voneinander getrennt werden. Ein Client kann gleichzeitig auf mehrere Zellen zugreifen, muß aber mindestens in einer Mitglied sein.

Zellen bestehen aus einer oder mehrerer Volumes oder auch Volumeinstanzen.

Auf dem Client wird ein Puffer für die Datenübertragung eingerichtet, der für eine Entlastung des Netzwerkes sorgt. Programme auf dem Client kommunizieren ausschließlich mit dem Puffer, der nur mit dem Server kommuniziert. Dadurch wird im Falle eines Netzwerkfehlers sichergestellt, daß der Anwender einfach weiterarbeiten kann. Ist der Server wieder erreichbar, gleicht der Puffer automatisch seinen Datenbestand mit dem Server ab.

Ein AFS Server verlangt vom Client eine Anmeldung über eine zentrale Instanz. Ohne eine Anmeldung kann der Client nur auf öffentliche Dateien und Verzeichnisse zugreifen. Dadurch wird sichergestellt, daß ein kompromittierter Client kein Sicherheitsproblem für das AFS wird, da durch die zentrale Instanz die Nutzerrechte der AFS Daten bis zu den Clients gültig sind.

Es gibt im OpenAFS folgende Dateiberechtigungen:

- Inhalt von Dateien lesen (**r**ead)
- Dateien schreiben (**w**rite)
- Verzeichnisinhalt auf**l**isten
- Dateien anlegen (**i**nsert)
- Dateien löschen (**d**elete)
- Dateien sperren (loc**k**)

- Zugriffsrechte **a**dministrieren

Diese Berechtigungen werden auf Verzeichnisse angewendet und schließen die enthalten Dateien und Unterverzeichnisse mit ein. Einzelnen Dateien in einem Verzeichnis können keine gesonderten Attribute zugewiesen werden.

Es gibt verschiedene Benutzergruppen, über die der Zugriff auf die Daten geregelt wird. Dazu zählen:

- `system:anyuser` Alle Benutzer, auch die unangemeldeten
- `system:authuser` alle angemeldeten Benutzer der lokalen Zelle
- `system:administrator` alle Administratoren

Vom Anwender können Gruppen erzeugt und Benutzer zu den erzeugten Gruppen hinzugefügt werden.

UNIX und Windows Dateirechte werden vom AFS ignoriert und sollten nicht verwendet werden.

Der AFS Client unter Windows bietet einen Eintrag im Kontextmenü des AFS Verzeichnisses, unter dem die AFS Rechte eingesehen und verändert werden können.

Unter UNIX werden die AFS Dateirechte über das Kommandozeilenprogramm `/usr/bin/fs` verwaltet. Hier einige Beispiele für typische Kommandos der Rechteverwaltung:

- `fs listacl verzeichnis` auflisten der Rechte von *verzeichnis*
- `fs setacl verzeichnis abcuser rwi` Der Benutzer abcuser bekommt für das Verzeichnis *verzeichnis* das Recht, Dateien zu lesen und zu schreiben und darf neue Dateien und Verzeichnisse erzeugen
- `pts creategroup abcuser:xyz` Gruppe abcuser:xyz anlegen
- `pts adduser xyzuser abcuser:xyz` Benutzer xyzuser der Gruppe abcuser:xyz hinzufügen

5.1.3 besondere Eigenschaften

Volumes lassen sich im laufenden Betrieb auf einen anderen Server verschieben. Somit wird der Nutzerbetrieb durch eine Wartung am Server nicht beeinträchtigt.

Von Volumes können physikalische Kopien auf mehrere Server verteilt werden, die nur lesbar sind und sich automatisch mit dem Original abgleichen. Dadurch wird Ortsunabhängigkeit und Redundanz der Daten im Falle eines Serverausfalls sichergestellt.

Auf Daten wird vom Client über eine URL zugegriffen (Ortunabhängiger Dateizugriff). Dadurch kann im Falle eines Performance-Engpass einfach ein neuer Server in das AFS integriert werden, aber die Art des Zugriffs auf die Daten ändert sich nicht.

Die aktuelle Implementierung von OpenAFS konzentriert sich auf folgende Server Plattformen:

- Linux
- AIX
- Solaris

AFS ist für folgende Betriebssysteme als Client verfügbar:

- Linux
- AIX
- Solaris
- MacOS X
- Windows

Mehr zum Thema unter openafs.org [2009].

5.2 CIFS/SMB

SMB ist die Abkürzung für Server Message Block und CIFS für Common Internet File System. CIFS ist eine Weiterentwicklung von SMB und nutzt als Grundlage *NetBIOS over TCP/IP*.

5.2.1 Herkunft

SMB wurde im Jahr 1983 von Barry Feigenbaum bei der Firma IBM entwickelt. Das Common Internet File System wurde 1996 von der Firma Microsoft eingeführt.

5.2.2 Funktionsweise

Das Netzwerkdateisystem basiert auf *NetBIOS*, welches wiederum als Grundlage *NetBEUI* oder *TCP/IP* verwendet. Da in dieser Arbeit nur das Dateisystem SMB/CIFS betrachtet wird, werde ich mich auf NetBIOS mittels TCP/IP beschränken.

Details zum Protokoll NetBIOS über TCP/IP oder kurz NBT sind in den RFCs 1001 und 1002 zu finden.

Die freie Implemetierung des SMB Protokolls heißt *Samba* und wurde 1994 von Andrew Tridgell als Open Source Projekt gegründet.

Samba stellt in einem Netzwerk Datei- und Druckdienste zur Verfügung. Es arbeitet nach dem Client-/Server-Modell, wobei ein Server Ressourcen wie Speicherplatz und Drucker in einem Netzwerk freigibt. Clienten, die sich bei dem Samba Server authentifizieren, können nun mit den Ressourcen arbeiten.

Unter Samba gibt es zwei *Daemonen* die für den Betrieb wichtig sind:

- smbd - Verantwortlich für Freigabe von Druckern und Verzeichnissen, Authentifiezierung
- nmbd - Namensauflösung

5.2.2.1 smbd

Der smbd Daemon bietet zwei verschiedene Sicherheitsmodelle für Ressourcen zur Verwendung an. Die erste *share level security*, beschränkt den Zugriff je Freigabe mit einem Passwort. Es gibt keine Benutzer im klassischen Sinn, sondern jeder der das Passwort für die Freigabe kennt, kann darauf zugreifen.

Die zweite, *user level security*, beschränkt den Zugriff auf Ressourcen nach Benutzern und Gruppen. Jeder Client muss sich gegenüber den Samba Server authentifizieren, bevor er Ressourcen benutzen kann. Die Authentifikation kann entweder unverschlüsselt oder verschlüsselt erfolgen.

Von der Verwendung mit unverschlüsselten Passwörtern wird abgeraten.

Das BSI [2010b] schreibt dazu:

„Da Klartext-Passwörter mit Hilfe von frei zugänglichen Tools beim Transport über das Netz leicht abgehört werden können, sollten grundsätzlich nur verschlüsselte Passwörter zum Einsatz kommen. “

Um die Verschlüsselung von Passwörter zu erzwingen, sind folgende Einträge in die Datei `smb.conf` einzutragen:

Listing 5.1: Verschlüsselte Passwörter in Samba

```
1 encrypt password = yes
2 smb passwd file = /etc/samba/smbpasswd
```

Der Server verlangt nun vom Client eine NTLMv2 Authentifizierung. Wie diese Authentifizierung im Detail funktioniert ist im MSDN [2010] nachzulesen. Die Verschlüsselung der Nutzdaten ist mit Samba nicht möglich.

Um Lese- und Schreibvorgänge im Netzwerk zu beschleunigen wird auf dem Client ein Cache eingerichtet. Um Kohärenzprobleme zu vermeiden bietet der smbd die Möglichkeit die Verwaltungsstrategie des Caches dynamisch zu verändern.

5.2.2.2 nmbd

TCP/IP verwendet 4 Byte lange Adressen um Rechner zu identifizieren. So wird beispielsweise ein Server mit der IP Adresse `192.168.0.1` angesprochen. Unter NetBIOS werden Rechner konsequent mit ihren 25 stelligen Rechnernamen adressiert. Jeder Rechner in einem Netzwerk besitzt eine eindeutige IP Adresse und einen Hostnamen. NetBIOS bietet zwei Möglichkeiten um Hostnamen und IP Adressen in einem Netzwerk bekannt zu machen.

Die erste Möglichkeit wird über einen Rundruf im lokalem Netzwerk realisiert. Die Kommunikation per Rundruf ist immer an die lokale Rundruf Adresse, oder *Broadcast* Adresse gerichtet. Für ein Netzwerk mit der Netzwerkadresse `192.168.0.0` und der Netzmaske `255.255.255.0` lautet die Rundruf Adresse `192.168.0.255`. Datenpackete die an diese Adresse adressiert sind, werden von allen Hosts empfangen. Möchte beispielsweise RechnerA mit RechnerB Daten austauschen, fragt RechnerA im Netzwerk: „Wer ist RechnerB“? Und RechnerB antwortet: „Ich bin es“. Somit kennt RechnerA die IP Adresse von RechnerB und kann nun eine TCP/IP Verbindung an den smbd Dienst auf RechnerB aufbauen.

Die zweite Möglichkeit wurde mit dem Hintergedanken entwickelt, daß regelmässige Rundrufe in einem lokalem Netzwerk eine nicht zu vernachlässigende Belastung für die Infrastruktur verursacht, da alle Rundruf Pakete von allen Netzwerkteilnehmern der Rundruf-Domäne empfangen und ausgewertet werden müssen.

Es wurde ein Dienst mit dem Namen NetBIOS Name Service, kurz NBNS realisiert, der alle Rechnernamen und IP Adressen im lokalem Netzwerk kennt. Dazu muss sich jeder Host beim Systemstart an dem NBNS registrieren. Dabei wird nicht nur der Hostname und die IP Adresse registriert, sondern auch ob der Registrant Druck- oder Datei-Dienste im lokalem Netzwerk zur Verfügung stellt.

Dadurch wird dem Anwender ermöglicht, alle Druck- und Datei-Dienste im lokalem Netzwerk aufzulisten. In der NetBIOS Welt wird diese Eigenschaft browsen genannt. Möchte nun RechnerA mit RechnerB Daten austauschen, fragt RechnerA den NBNS, welche IP Adresse RechnerB besitzt. Der NBNS

ist nicht zu verwechseln mit dem *DNS* unter TCP/IP. Im Gegensatz zum *DNS* ist die Topologie nicht hierarchisch sondern flach und auf kleine Netzwerke ausgelegt. Dieser Dienst wird unter Samba von dem Daemon **nmbd** realisiert.

Mehr zu dem Thema ist unter Smith [2001] Seite 17 zu erlesen.

Am 1. März 2010 wurde die Version 3.5.0 von Samba veröffentlicht. Eine experimentelle Implementierung des neuen Protokolls SMB2 wurde in der neuen Version eingebaut.

5.3 Coda

5.3.1 Herkunft

Coda wurde 1987 an der Carnegie Mellon University in Pittsburgh, USA als Nachfolger von AFS entwickelt.

5.3.2 Funktionsweise

Der Aufbau und die Funktionsweise von Coda sind dem Vorgänger *AFS* ähnlich. Die zwei Stärken von Coda liegen in *disconnected Operation* und *Server Replication*. Daran sind schon die Designziele des Netzwerkdateisystems zu erkennen. Neben guter Skalierbarkeit und Sicherheit ist das vor allem gute Verfügbarkeit. Das Coda Dateisystem ist tolerant gegenüber Netzausfällen. Diese Eigenschaft macht Coda für den Einsatz auf mobilen Computern interessant.

Coda funktioniert nach dem Client/Server-Modell, bei dem ein oder mehrere Server Daten anbieten und Clienten auf diese über ein Netzwerk zugreifen. In der Coda-Welt wird der Server als sogenannter Vice und der Client als Venus Prozess bezeichnet. Diese Prozesse sind als Menge von Nebenläufigen Threads organisiert, die ausschliesslich im Benutzerkontext laufen.

Coda setzt auf *RPC2* auf, was als Nachfolger von *RPC* einige Vorteile hat. RPC2 bietet einen kryptografischen Schlüsselaustausch und Authentifikation der Kommunkationspartner basierend auf dem Algorithmus *Needham and*

Schroeder. Somit werden Man-in-the-Middle Attacken verhindert und die Kommunikationspartner eindeutig identifiziert. Weiterhin unterstützt RPC2 Multicast auch MultiRPC genannt, um z.B. vom Server Ungültigkeitsnachrichten an alle Clients, die eine Datei geöffnet haben, zu senden. Diese Funktion wird zur Verhinderung von Kohärenzproblemen von Coda verwendet.

Unter Coda werden die Daten unter einem globalen Namensraum angeboten und sind somit unabhängig von physikalischen Standort, Plattform oder Betriebssystem des Servers immer unter der gleichen URL erreichbar.

Eine URL an der Hochschule Mittweida könnte z.B. so aussehen:

`/coda/minic.htwm.de/mni/benutzername`

Dabei wird in der Coda-Welt der Ordner `mni` als Zelle bezeichnet. Da Coda eine Weiterentwicklung von OpenAFS ist, sind einige Eigenschaften wie bei seinem Vorgänger.

Zellen bestehen aus einer oder mehrerer Volumes oder auch Volumeinstanzen. Diese werden jetzt physiklischen Coda-Servern zugewiesen, die dann eine Volume Storage Group (VSG) bilden. Öffnet ein Client eine Datei, lädt er sie von einem Server aus der VSG in seinen lokalen Cache. Wird die Datei geschlossen, wird mithilfe von MultiRPC die neue Datei an alle Server im VSG gesendet.

Ein Coda Anwender greift auf das eingebundene Coda Laufwerk mittels der URL **/coda/minic.htwm.de/mni/benutzername** zu. Dabei spielt es keine Rolle, ob der Client mit den Server verbunden ist oder nicht. Es ist ebenso unwichtig, auf welchem Volume die Dateien liegen oder ob es ein repliziertes Volume ist. Der Zugriff auf die Dateien erfolgt immer über diesen Pfad. Dadurch wird echte Ortunabhängigkeit erreicht.

5.3.3 besondere Eigenschaften

Coda ist noch immer ein Forschungsprojekt an der CMU und wird kontinuierlich weiterentwickelt.

- replizierte Volumes um Server näher am Client aufzustellen und die Latenz zu verringern

- Schutz vor Man-in-the-Middle Angriffen
- skalierbar
- robust gegen Netzwerkfehler

Ein wesentlicher Nachteil des Coda Dateisystems ist der hohe Administrations und Konfigurationsaufwand.

Der Windows Client ist seit Februar 2008 im *Beta Status* und hat keine Änderungen erhalten. Die Entwickler konzentrieren sich auf die Betriebssysteme FreeBSD und Linux. Der Windows Client wurde aus diesem Grund nicht weiter betrachtet.

5.4 NFSv3

Um einen direkten Vergleich mit dem abzulösenden Netzwerkdateisystem zu ermöglichen, wird NFSv3 betrachtet. NFS ist die Abkürzung für *Network Filesystem*, zu deutsch Netzwerkdateisystem.

5.4.1 Herkunft

NFS ist ist ein von der Firma Sun Microsystems entwickeltes Protokoll, um auf Daten eines entfernten Servers zuzugreifen. NFS wurde mit SunOS2.0 1985 in der zweiten Version veröffentlicht. Diese zweite Version von NFS unterstützt nur eine maximale Dateigrösse von 4GB und UDP als Transportmedium. Seit 1995 ist die Version 3 von NFS verfügbar, in der 64-Bit Dateigrößen, ein asynchroner Betrieb und TCP als Übertragungsprotokoll unterstützt wird.

5.4.2 Funktionsweise

Im Unix Umfeld ist das *NFS* eine der meist verbreitesten verteilten Dateisysteme. NFS arbeitet nach dem klassischen Client-Server Modell mit der Besonderheit, eines statuslosen Dateisystems.

Ein statusloses Dateisystem speichert keine Daten von Clienten auf dem Server, sondern es werden Authentifizierungs- und Dateiinformationen mit jeder Anforderung übertragen. Nachteilig an diesem Modell ist die mehrfache Datenübertragung von gleichen Informationen.

Die POSIX Spezifikation definiert die Möglichkeit, Dateien mittels **lockf()** zu sperren. Möchte ein Client in eine Datei schreiben, muss er die Ressource vorher sperren, damit kein zweiter Client auf diese Ressource schreibenden Zugriff bekommt.

Doch wie implementiert ein NFS-Server Status behaftete Dienste wie das Sperren einer Ressource, in einem statuslosen Client Server Modell ?

Dazu schreibt P.A.Gloor [1989] auf Seite 162 Kapitel 6:

„Beim einfachsten Ansatz führt der Server die globale Lock-Tabelle, wobei die Übereinstimmung der Einträge in der Lock-Tabelle mit dem tatsächlichen Zustand der Clients durch das Locking-Protokoll sichergestellt werden muss."

Der Server muss sich die Zustände in einer Lock-Tabelle speichern. Somit wird das Konzept eines statusbehafteten Servers wieder auf Benutzerebene eingeführt. Diese Funktionalität wurde von Sun Microsystems mit dem Daemon `rpc.lockd` und dem Protokoll *NLM* implementiert. Problematisch bei diesem Ansatz ist, daß ein abgestürzter Client eine Ressource blockieren kann.

Also wird ein weiterer Dienst benötigt, der den Status der Clienten sequenziell überwacht und an den Server meldet. Dieser Dienst wird von dem Daemon **rpc.statd** implementiert.

Es ist ersichtlich, daß NFSv3 ein Zusammenschluss von Teilprotokollen, mit denen die Kommunikation zwischen Client und Server geregelt wird, ist. Dazu wird *RPC* verwendet, der über den Daemon **portmapper** dynamisch Kommunkationsports festlegt. Die nachfolgende Auslistung zeigt die geöffneten Ports eines typischen NFS Servers.

Listing 5.2: offene Ports NFSv3

```
1 COMMAND PID USER FD  TYPE DEVICE SIZE NODE NAME
2 portmap  2170 daemon  6u  IPv4  5829       UDP *:sunrpc
```

```
portmap  2170 daemon  7u  IPv4  5832  TCP *:sunrpc (LISTEN)
rpc.statd 2345  statd  6u  IPv4  6106  UDP *:825
rpc.statd 2345  statd  8u  IPv4  6114  UDP *:55352
rpc.statd 2345  statd  9u  IPv4  6117  TCP *:46405 (LISTEN)
rpc.mount 2558  root   6u  IPv4  6456  UDP *:49311
rpc.mount 2558  root   7u  IPv4  6461  TCP *:45476 (LISTEN)
```

Für einen derartigen Server lassen sich nur schwer die passenden Firewall Regeln erstellen und verwalten. Das ist ein wesentlicher Kritikpunkt an NFSv3.

Die Nutzer werden unter NFSv3 mittels *MNTV3* Protokoll authentifiziert. Dieser Dienst, der durch den **mountd** Daemon realisiert wird, überprüft die Identität des Benutzers und prüft die Zugriffsrechte. Dabei sendet der Client an den Server die gewünschte *UID* und *GID* und der Server prüft diese gegen die Dateirechte auf der Platte. Das Rechtemodell basiert somit auf einer Vertrauensstellung zwischen Rechnern basierend auf IP-Adressen. Dieses Modell gilt heutzutage als unsicher.

Dazu schreibt das BSI [2010a] : „Für die Verwendung der Protokolle NFS für den Export von Dateisystemen und die Verteilung von Systemdateien mittels NIS sind keine ausreichenden Schutzmaßnahmen in geschützten Umgebungen verfügbar. Der Einsatz stellt somit eine Gefährdung der Integrität der Systeme dar.“

Auch das ist ein wesentlicher Kritikpunkt an NFSv3, weshalb an der Hochschule Mittweida nach einer Alternative gesucht werden soll.

5.4.3 besondere Eigenschaften

NFS ist eines der verbeitesten Netzwerkdateisysteme und für viele Plattformen und Systeme gibt es Client- oder Server-Software. Die Software ist ausgereift und entsprechend getestet. Durch das zustandslose Protokoll ist NFS relativ geschützt gegen Abstürze von Server oder Clienten. NFS in der Version 3 verwendet UDP als Standardeinstellung, kann aber auf TCP umgestellt werden.

5.5 NFSv4

NFS ist die Abkürzung für *Network Filesystem*, zu deutsch Netzwerkdateisystem. NFSv4 ist der Nachfolger von NFSv3.

5.5.1 Herkunft

Seit August 2002 wird an der Universität von Michigan (USA) die Referenzimplementierung von NFSv4 für den Linux Kernel entwickelt. Seit der Kernel Version 2.6.4 ist der entwickelte Quellcode offizieller Bestandteil des Linux Kernel.

Die Weiterentwicklung von NFS hat dabei folgende Ziele:

- verbessertes Sicherheitsmodel
- Optimierung des Protokolls für das Internet
- Steigerung der Performance
- einfache Integration zukünftiger Erweiterungen

5.5.2 Funktionsweise

NFSv3 war ein Zusammenschluss von Teilprotokollen wie `rpc.mountd`, `rpc.lockd`, `rpc.statd` , `NLM`, `ACL` und `NFS`. In der neuen NFS Version 4 wurden alle Teilprotokolle integriert und unter dem RFC 3530 spezifiziert. Die komplette Kommunikation mit Clienten verläuft nun über den TCP Port 2049. Dadurch wird die Erstellung von Firewall-Regeln deutlich erleichtert und die Möglichkeit geschaffen, z.B. NFS über eine *SSH* Verbindung zu tunneln. Die Kommunikation über das Internet wurde damit deutlich verbessert.

Die neue Version von NFS ist auch nicht mehr zustandslos, sondern es gibt Sitzungen. Der Client muss sich am Server anmelden und bekommt von diesem eine eindeutige Identifikationsnummer. Der Server verwirft nach einer Ablaufzeit `leasetime` diese Sitzung, falls der Client sich nicht mehr meldet.

Durch die Sitzungsverwaltung weiß der Server, welche Clienten an einer Datei arbeiten. Ändert ein Client diese Datei, kann der Server alle anderen Clienten veranlassen, die Datei neu in den Puffer zu laden.

In NFSv3 mussten, um eine Datei zu lesen drei RPC Aufrufe an den Server gesendet und die entsprechende Antwort gelesen werden. Mit NFSv4 können RPC Kommandos mit einem *Compound remote procedure call*, einer Vereinigung vieler Aufrufe in einem Kommando, an den Server geschickt werden und dadurch die Signallaufzeit durch das Netzwerk und den Treiber gespart werden.

In der neuen NFS Version wurde die Interpretation der *Access Controll Lists*, kurz *ACL* dahingegend optimiert, daß sie voll POSIX und Windows kompatibel sind. Die Eigenschaften der Zugriffskontrolle werden in Form von UTF-8 kodierten Strings gespeichert, wodurch die ACL mit dem Windows Explorer verwaltet werden kann.

5.5.3 besondere Eigenschaften

Eine NFSv4 Installation kann neben einer bestehenden NFSv3 parallel betrieben werden. Somit kann der Umstieg von der alten zur neuen NFS Version zeitlich geordneter erfolgen.

Derzeitig werden an dem NFS Server die physikalischen Dateisysteme einzeln exportiert und müssen am Client alle einzeln eingebunden werden.

Dazu die gekürzte Ausgabe des Kommandos `mount` auf dem Rechner **hercules.mni.hs-mittweida.de**:

Listing 5.3: mount

```
1 minoc.htwm.de:/home/minoc27 on /home/minoc27 type nfs
2 minoc.htwm.de:/home/minoc49 on /home/minoc49 type nfs
3 minoc.htwm.de:/home/minoc16 on /home/minoc16 type nfs
4 minoc.htwm.de:/home/minoc38 on /home/minoc38 type nfs
5 minoc.htwm.de:/home/minoc05 on /home/minoc05 type nfs
6 minoc.htwm.de:/home/minoc on /home/minoc type nfs
```

Mit der neuen NFS Version muss der Client nur noch ein NFS exportiertes Verzeichnis in seinen Verzeichnisbaum einbinden und kann dann rekursiv in alle Unterverzeichnisse des Server zugreifen.

In der NFS Version 4.1 wurde im Standard eine Funktion implementiert, die sich pNFS nennt. Damit ist es möglich, daß ein NFS Server zum Metadatenserver wird. Mit dieser Funktion kann ein Cluster mit viel Bandbreite aufgebaut werden und NFS wird dadurch hoch skalierbar. Es können einzelne Dateien über mehrere NFS Server verteilt werden oder sogar der Zugriff auf einzelne Speicherblöcke verwaltet werden.

Implementiert wurde diese Funktion für die Betriebsysteme FreeBSD, Solaris und Linux. Mehr zu dem Thema ist unter NetAPP [2010] zu finden.

Das Potential von NFSv4 hat sogar die Firma Microsoft erkannt und hat die Universität von Michigan (USA) damit beauftragt, bis Ende des Jahres 2010 einen Client für Microsoft Betriebssysteme zu entwickeln. Mehr Informationen dazu sind auf der Homepage vom Citi [2010] zu finden.

Leider gibt es derzeitig keinen freien Verfügbaren NFSv4 Client für Windows. Es gibt eine kommerzielle Version der Firma hummingbird.com die unter dem Namen `Open Text NFS Client 14` vertrieben wird.

6 Praktische Betrachtung der verbliebenen Netzwerkdateisysteme

Das Ziel der praktischen Gegenüberstellung ist es, Aussagen über den Bedarf an Netzwerk- und Systemressourcen der betrachteten Dateisysteme zu treffen. Dazu werden die Tests der verschiedenen Dateisysteme auf der gleichen Hardware auf Server und Client stattfinden, um Unterschiede der Software zu messen.

Die praktische Implementierung soll auch Aussagen über die folgenden Fragen erlauben:

- kann das Dateisystem nach einer erfolgreichen Anmeldung am Clienten eingebunden werden
- wie Aufwendig ist die Implemetierung auf dem Client unter Linux oder Windows

6.1 Versuchsaufbau

Basis für den Aufbau ist ein 100 MBit Netzwerk, an dem ein Server und zwei Clienten mit Verkabelung vom Typ Kat5 UTP angebunden sind. Der Switch ist das Modell `AT-FS713FC/SC` der Firma Allied Telesyn, der ausschliesslich für den Test zur Verfügung steht. Dadurch wird der Test nicht durch unvorhersehbaren Netzwerkverkehr beeinflusst.

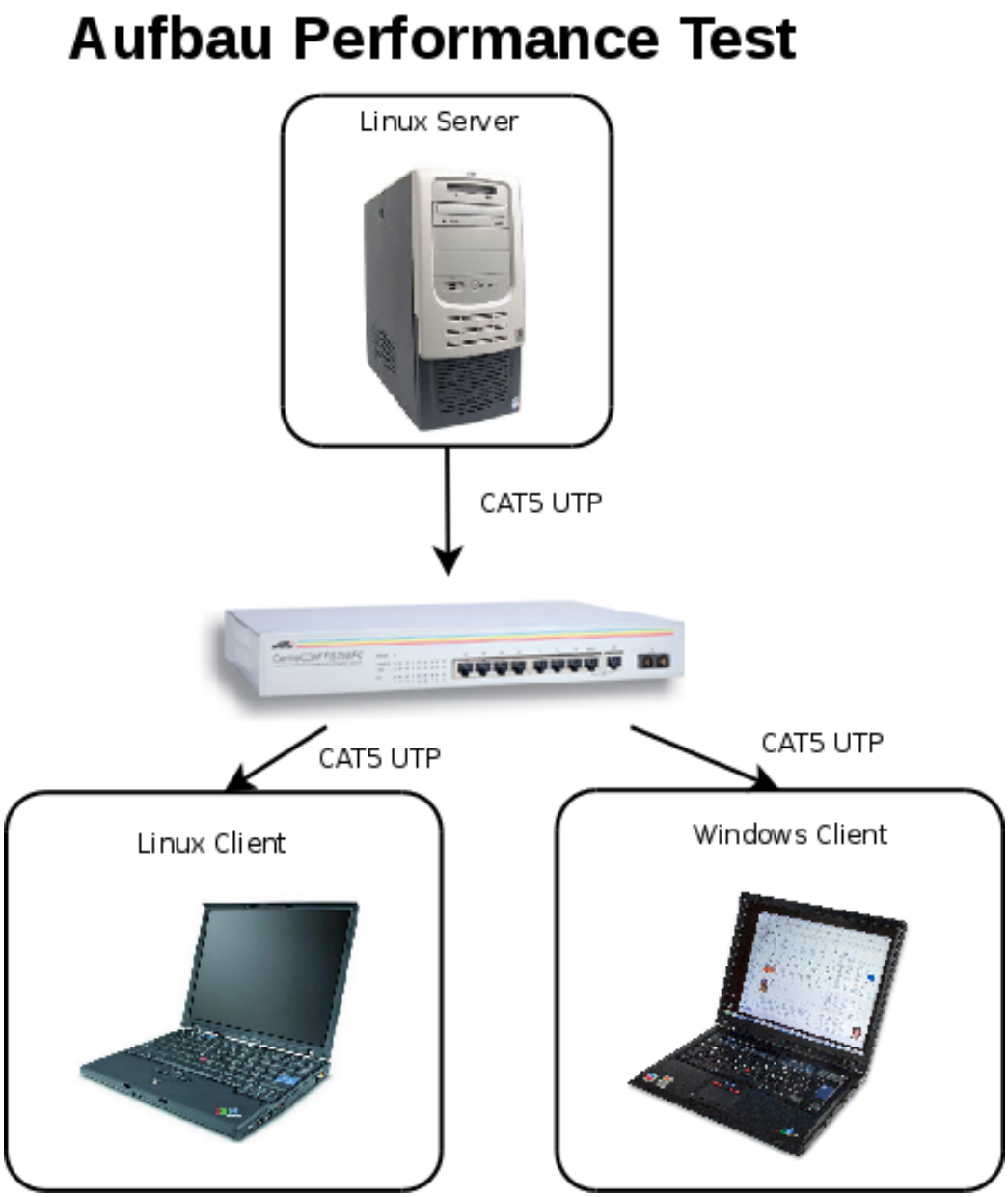

Abbildung 6.1: Aufbau Performancetest

Der Test wird mit der Software `iozone` durchgeführt, die für Linux und Windows kostenlos verfügbar ist. Getestet werden folgende Dateisystemoperationen:

- lesen
- wieder lesen
- schreiben

- wieder schreiben

Es wurde mit zwei verschiedenen Dateigrößen 4 KByte und 512 MByte gemessen. Das komplette Kommando lautet:

```
iozone -Rab /tmp/out.xls -i 0 -i 1 -f /mnt/dump -q 4k -n 512M -g 512M
```

Die Kommandozeilenoptionen bedeuten dabei im Einzelnen:

- **-R** generiere einen Excel Report
- **-a** automatischer Testmodus
- **-b** Dateiname der Exceldatei
- **-i** Einstellung der Testart (Lese,Wiederlesen ...)
- **-f** Dateiname, auf dem die Tests durchgeführt werden
- **-q** Größe der Blöcke einer Operation, read() oder write(), in KByte
- **-n|-g** Größe der Testdatei

Mehr zu dem Programm ist unter iozone.org [2010] zu finden.

6.1.1 Server

Der Server ist ein `HP Vectra VL` mit einem 500MHz Pentium 3 Prozessor und 192 MB SD Arbeitsspeicher. Die Netzwerkkarte ist eine `Intel Pro 82540EM` 1000 Mbit PCI.

Da die Implementierung der virtuellen Maschine mit dem Betriebssystem Debian GNU/Linux 5.0 realisiert werden soll, werden die Tests mit dem selbigen Betriebssystem durchgeführt.

Der Server verfügt über ein CD Laufwerk, so daß die Installation des Betriebssystems von einer CD erfolgen konnte. Zum aktuellen Zeitpunkt ist Version 5.0.4 von Debian verfügbar.

Der Debian Installer bietet während der Installation eine Vielzahl von Einstellungen an. Die 4.3 GB Festplatte wurde vom Installer in eine 250 MB Swap Partition und eine 4 GB Rootpartition geteilt.

Es wurde eine Installation gewählt, bei der nur die Basissoftware des Betriebssystems installiert wurden. Unter Debian 5.0.4 wurde die Linux Kernel Version 2.6.26-2 installiert.

Nach der Installation des Betriebssystems, wurde die Software `hdparm` installiert, um die Datenrate der Festplatte zu messen.

Dazu wurde ein Lesetest durchgeführt:

```
hdparm -t -direct /dev/hda
```

Listing 6.1: Leseperformance

```
1 /dev/hda:
2  Timing O_DIRECT disk reads: 32 MB in 3.03 seconds = 10.55 MB/sec
```

Die Kommandozeilenoption `-direct` veranlasst das Programm, den Kernel Page Cache zu umgehen und direkt mittels `raw IO` von der Festplatte zu lesen.

Das bedeutet im Ergebnis, daß von der Festplatte mit einer Bandbreite von 10 MByte/Sekunde ohne Dateisystem gelesen werden kann.

6.1.2 Client

Der erste Client ist ein IBM Thinkpad R51 Notebook mit Windows XP SP2 Betriebssystem, einen Intel Pentium M Prozessor mit 1,7 GHz und 512 MByte SD Arbeitsspeicher. Die Netzwerkkarte ist eine Intel 1000 MT Pro mit Intel Treiber Version 7.2.17.101.

Der zweite Client ist ein IBM Thinkpad X61s Notebook mit Debian GNU/Linux 5.0.4 Betriebssystem, einen Intel Core 2 Duo Prozessor mit 1.6 GHz und 2GByte DDR1 Arbeitsspeicher. Die Netzwerkkarte ist eine Intel 82566MM mit Intel Treiber Version 1.0.2-k2 (Linux Kernel 2.6.33).

6.2 NFSv3

6.2.1 Einrichten des Servers unter Debian GNU/Linux

Der Server wurde mit der Debian GNU/Linux Minimalinstallation installiert um einen definierten, initialen Zustand für den Test zu haben. Die Debian Paketverwaltung bietet zwei Möglichkeiten an, um einen NFS Server zu implementieren.

Die erste Möglichkeit beinhaltet einen NFS Server im Usermode und die zweite im *Kernelmode*.

Mit dem Kommando `aptitude install nfs-kernel-server` wurde der NFS Server, der im Kernelmode läuft, von der Paketverwaltung installiert und konfiguriert.

In der Datei `/etc/exports` wird folgender Eintrag eingefügt:

Listing 6.2: /etc/exports

```
1 /home linux(rw,sync,no_subtree_check) windows(rw,sync,no_subtree_check)
```

Der Eintrag bewirkt, daß das Verzeichnis `/home` für den Rechner **linux** und **windows** im Netzwerk freigegeben wird. Die beiden Clienten können auf das Verzeichnis lesend wie auch schreibend zugreifen.

Listing 6.3: /etc/hosts

```
1 192.168.3.115  linux
2 192.168.3.116  windows
```

Nachdem der NFS Server neugestartet wurde, ist der NFSv3 Server fertig konfiguriert.

Durch das Kommando `showmount -e 192.168.3.135` werden die exportierten Verzeichnisse des Rechners mit der IP 192.168.3.135 angezeigt.

Listing 6.4: showmount -e

```
1 Export list for 192.168.3.135:
2 /home windows,linux
```

6.2.2 Einrichten des Clients unter Windows XP SP2

Der Hersteller des Betriebssystems Windows stellt unter dem Namen `Microsoft Services for UNIX` ein Programmpacket zur Verfügung, in dem ein NFS Client enthalten ist.

Das Programm kann unter der URL

`http://technet.microsoft.com/de-de/interopmigration/bb380242.aspx`

kostenfrei geladen werden.

Nach dem Herunterladen der 217.6 MByte großen Datei `SFU35SEL_EN.exe`, kann durch Doppelklick die Installationsroutine gestartet werden.

Zum Zeitpunkt dieser Arbeit wurde die Software in der Version **3.5** build **8.0.1969.1** angeboten.

Da NFSv3 eine Zugangskontrolle auf Daten mittels Benutzer- und Gruppenzugehörigkeit implementiert, müssen Windows Benutzer zu UNIX Benutzern auf dem NFS Server verknüpft werden.

Das kann über ein funktionierendes *NIS* Verzeichnis im Netzwerk erfolgen oder mit dem Dienst `User Name Mapping`, der auf dem Windows Client installiert und konfiguriert werden muss. Dazu müssen die Dateien `/etc/passwd` und `/etc/group` vom NFS Server auf dem Windows Client kopiert werden.

Jetzt müssen lokale Windowsbenutzer mit den zugehörigen UNIX Benutzern verlinkt werden. Dazu wurde ein Bildschirmausschnitt angefertigt, bei dem diese Zuordnung zu sehen ist.

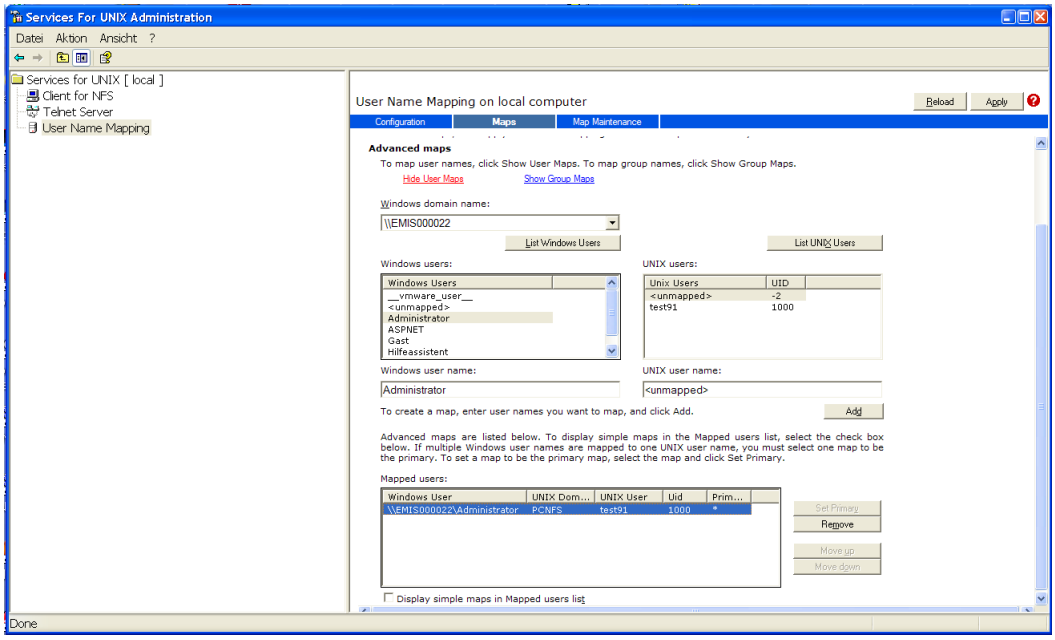

Abbildung 6.2: User Mapping auf einem NFSv3 Windows Client

Die Zuordnung von Windows- zu UNIX-Benutzern kann in größeren Netzwerken einen ziemlich großen Verwaltungsaufwand in Anspruch nehmen.

Nach dem typischen Neustart des Windows Rechners, kann auf den NFS Server zugegriffen werden. Die Installation ist damit abgeschlossen.

6.2.3 Einrichten des Clients unter Debian GNU/Linux 5.0

Die Debian Packetverwaltung bietet einen NFS Client in dem Packet `nfs-common` an.

Nach der Installation und den folgenden Eintrag in die Datei **/etc/fstab**

Listing 6.5: /etc/fstab

```
1 192.168.3.135:/home /mnt        nfs       defaults        0       0
```

wird der NFS Client installiert und konfiguriert. Mit dem Kommando

```
mount /mnt
```

wird das exportierte Verzeichnis vom NFS Server in den lokalen Verzeichnisbaum eingebunden.

6.3 CIFS/Samba

6.3.1 Einrichten des Servers unter Debian GNU/Linux

Mit dem Kommando

apitude install samba

wird der Samba/CIFS Server von der Packetverwaltung in der Version 3.2.5 installiert und eingerichtet. Die notwendigen Dienste wurden gestartet und warten auf Verbindungen von Clienten.

Listing 6.6: lsof -i

```
1 COMMAND PID  USER FD  TYPE DEVICE SIZE NODE NAME
2 nmbd    2415 root  8u  IPv4 6738       UDP *:netbios-ns
3 nmbd    2415 root  9u  IPv4 6739       UDP *:netbios-dgm
4 nmbd    2415 root 10u  IPv4 6741       UDP *:netbios-ns
5 nmbd    2415 root 11u  IPv4 6742       UDP *:netbios-dgm
6 smbd    2417 root 21u  IPv6 6790       TCP *:microsoft-ds (
    LISTEN)
7 smbd    2417 root 22u  IPv6 6792       TCP *:netbios-ssn (
    LISTEN)
```

Um die Tests durchzuführen, wurde eine minimale Konfiguration des Servers gewählt, in der anonyme Clienten Lese- und Schreibrechte auf ein Verzeichnis erhalten.

Listing 6.7: /etc/samba/smb.conf

```
1 [global]
2 workgroup = TEST
```

```
3  netbios name = TEST
4  security  = share
5
6  [data]
7  comment = test
8  path = /tmp
9  force user = nobody
10 force group = nogroup
11 read only = No
12 guest ok = Yes
```

Die Minimalkonfiguration wurde aus dem Buch Kühnel [2004] entnommen. Damit war die Installation und Einrichtung auf dem Server beendet.

6.3.2 Einrichten des Clients unter Windows XP SP2

Das Protokoll CIFS kann in jedem Windows Betriebssystem durch die Installation der Software **Client für Microsoft Netzwerke** nachinstalliert werden. Der Windows Client muss der Arbeitsgruppe **TEST** beitreten. Nach einem Neustart ist er Mitglied in der Arbeitsgruppe und kann sich in der Netzwerkumgebung die Arbeitsgruppen-Rechner anzeigen lassen und durch die freigegebenen Ressourcen browsen.

Die Einrichtung ist damit abgeschlossen.

6.3.3 Einrichten des Clients unter Debian GNU/Linux 5.0

Die Installation und Einrichtung der Samba Client Software wird mit dem Kommando

```
apitude install smbfs smbclient
```

unter Debian GNU/Linux erreicht. Das Programm wird in der Version 3.2.5 installiert. Mit dem Kommando `smbclient -L 192.168.3.135` können die exportierten Ressourcen des Samba Servers mit der IP Adresse 192.168.3.135 angezeigt werden.

Listing 6.8: smbclient -L Sambaserver

```
Domain=[TEST] OS=[Unix] Server=[Samba 3.2.5]

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       IPC Service (Samba 3.2.5)
        data            Disk      test
Domain=[TEST] OS=[Unix] Server=[Samba 3.2.5]

        Server               Comment
        ---------            -------
        EMIS000022
        TEST                 Samba 3.2.5

        Workgroup            Master
        ---------            -------
        TEST                 TEST
```

Durch folgenden Eintrag in die Datei `/etc/fstab` wird das exportierte Dateisystem des Servers nach einen Neustart in das lokale Dateisystem eingebunden.

Listing 6.9: /etc/fstab

```
//192.168.3.135/data /mnt smbfs defaults 0 0
```

Der Samba Client unterstützt folgende Authentifizierungsmechanismen:

- **keine** Authentifizierung als kein Benutzer
- **krb5** Kerberos Authentifizierung Verion 5
- **ntlm** NT Lanman Passwort Hash (Voreinstellung)
- **ntlmv2** NT Lanman Passwort Hash Version 2

6.4 OpenAFS

6.4.1 Einrichten des Servers unter Debian GNU/Linux

Die Benutzer werden unter OpenAFS auf Basis von Kerberos authentifiziert. Die Installation des Server wird mit folgenden Kommando ausgelöst: `aptitude install openafs-fileserver openafs-client openafs-krb5 openafs-dbserver`.

Unter OpenAFS läuft ein Teil des Programm im *Kernelmode*, was unter Linux in Form von Kernel-Modulen realisiert wird.

Das OpenAFS Kernelmodul wird durch folgende Kommandos installiert und geladen:

- `aptitude install module-assistant`
- `module-assistant prepare openafs-modules`
- `module-assistant build openafs-modules`
- `module-assistant install openafs-modules`
- `modprobe openafs`

OpenAFS verlangt ein spezielles Benutzerkonto im Kerberos, welches folgendermassen auf dem Kerberosserver angelegt wird:

- `kadmin.local`
- `addprinc -randkey -e des-cbc-crc:v4 afs`
- `ktadd -k /tmp/afs.keytab -e des-cbc-crc:v4 afs`

Eine neue Zelle wird unter OpenAFS mit dem Kommando `afs-newcell` und `afs-rootcell` angelegt.

Da Kerberos nur die Benutzer authentifiziert, müssen diese trotzdem noch auf dem OpenAFS Server angelegt werden, was mit dem Kommando `pts createuser Benutzer -id 500` realisiert wird.

Durch die folgenden Kommandos wird dem Benutzer ein Datenbereich zugewiesen:

- `vos create afs /vicepa user.benutzer -maxquota 0`

- `cd /afs/.linux.test/benutzer`
- `fs mkm benutzer user.benutzer`
- `chown . benutzer`
- `fs sa benutzer benutzer all`
- `fs sa benutzer system:authuser 1`
- `vos release user`
- `fs checkvolumes`

Die Einrichtung des Server ist damit abgeschlossen.

6.4.2 Einrichten des Clients unter Windows XP SP2

Vor der Installation des OpenAFS Client muss der Kerberos Client installiert werden.

Der OpenAFS Client verlangt unter Windows die Installation des **Client für Microsoft Netzwerke**. Um die Systemzeit mit dem Zeitserver `time.linux.test` zu synchronisieren, wird das nachfolgende Kommando benutzt: `net time "/setsntp:time.linux.test"`

Der OpenAFS Client ist derzeitig in der Version 1.5.72 verfügbar und kann von der Webseite **http://openafs.org** heruntergeladen werden.

Wenn ein gültiges Token vom Kerberos Server für den AFS Client verfügbar ist, kann er auf den AFS Server zugreifen und einen AFS-Pfad mit einem Laufwerksbuchstaben verknüpfen.

6.4.3 Einrichten des Clients unter Debian GNU/Linux 5.0

Unter Linux wird der OpenAFS Client mit dem Kommando `aptitude install openafs-client` installiert. Um den Client gleich sinnvoll zu konfigurieren, stellt das Installationsprogramm einige Fragen:

- Wie heißt die AFS Zelle ?
- Wie groß soll der lokale Cache sein ?

- Soll der Inhalt von `/afs` dynamisch generiert werden ?
- Welches Hosts sind DB Server in der AFS Zelle ?
- Starten des AFS Client beim Systemstart ?

Nach der Beantwortung dieser Fragen ist der OpenAFS Client bereits fertig konfiguriert.

Da ein Teil des OpenAFS Programm im Kernelmode läuft, muss auch auf dem Client das Kernel Modul installiert werden:

- `module-assistant prepare openafs-modules`
- `module-assistant build openafs-modules`
- `module-assistant install openafs-modules`
- `modprobe openafs`

Mit dem Kommando `kinit Benutzername@linux.test` wird ein Kerberos Ticket vom Server beantragt. Um nun auf die AFS Resourcen zuzugreifen, muss noch `aklog -c linux.test` aufgerufen werden.

Die AFS Verzeichnisse sind nun unter `/afs/.linux.test/Benutzername` erreichbar. Es muss kein Verzeichnis manuell in den Verzeichnisbaum eingehangen werden, sondern es sind unter `/afs/.linux.test/` nur die Verzeichnisse erreichbar, die auch für den Benutzer freigegeben wurden.

6.5 Coda

6.5.1 Einrichten des Servers unter Debian GNU/Linux

Die Benutzer werden unter Coda auf Basis von Kerberos authentifiziert.

Die Software ist nicht im Standard-Verzeichnis von Debian GNU/Linux enthalten. Es muss eine externe Quelle zur Packetverwaltung hinzugefügt werden. Dazu wird folgende Zeile in die Datei `/etc/apt/source.list` eingefügt:

```
deb http://www.coda.cs.cmu.edu/debian stable/
```

Durch das Kommando `aptitude update` werden die lokalen Packetinformationen mit den Quellen synchronisiert.

Jetzt kann mittels `aptitude install code-server` die Software installiert werden.

Die Entwickler von Coda empfehlen, die exportierten Daten in einer expliziten Partition abzulegen. Diese sollte nicht mit einem Journaling-Dateisystem formatiert sein und unter `/vicepa` im Verzeichnisbaum eingebunden sein.

Mit der Software wurde ein interaktives Skript, `vice-setup` installiert, mit dem der Server konfiguriert wird.

Nach ausführen des Skriptes wurde eine neue `/etc/coda/server.conf` erstellt. In die Datei müssen noch die passenden Kerberos Parameter eingetragen werden.

6.5.2 Einrichten des Clients unter Debian GNU/Linux 5.0

Die Software ist nicht im Standard-Verzeichnis von Debian GNU/Linux enthalten.

Dazu wird folgende Zeile in die Datei `/etc/apt/source.list` eingefügt:

```
deb http://www.coda.cs.cmu.edu/debian stable/
```

Durch das Kommando `aptitude update` werden die lokalen Softwareinformationen mit den Quellen synchronisiert. Jetzt kann mittels `aptitude install code-client` die Software installiert werden.

Der Coda Client wird mit dem Skript `venus-setup` konfiguriert. Es werden zwei Kommandozeilenargumente erwartet:

- Hostname des Server
- Anzahl der Speicherblöcke des lokalen Coda Puffers

In dem Test wurde der Client mit folgenden Argumenten konfiguriert: `venus-setup coda.linux.test 20000`

Die Clientsoftware muss mittels `/etc/init.d/coda-client restart` neugestartet werden.

Durch das Kommando: `clog Benutzername@linux.test` kann sich der Anwender ein Kerberos Ticket holen.

Jetzt stehen unter dem Verzeichnis `/coda/.linux.test/Benutzername` seine Daten zur Verfügung.

6.6 NFSv4

6.6.1 Einrichten des Servers unter Debian GNU/Linux

Wie unter NFSv3 gibt es auch unter NFSv4 zwei Möglichkeiten den NFS Server zu betreiben. In dem Test wurde der NFS Server im Kernelmode verwendet. Mit dem Kommando `aptitude install nfs-kernel-server` wurde der NFS Server von der Packetverwaltung installiert und konfiguriert.

In der Datei `/etc/exports` wird folgender Eintrag eingefügt:

Listing 6.10: /etc/exports

```
2 /home gss/krb5(rw,sync,fsid=0,no_subtree_check,crossmnt)
```

Durch den Eintrag wird das lokale Verzeichnis `/home` exportiert.

Die Einrichtung des Server ist damit abgeschlossen.

6.6.2 Einrichten des Clients unter Debian GNU/Linux 5.0

Die Clientsoftware wird unter Debian GNU/Linux mit dem Kommando `aptitude install nfs-client krb5-user` installiert.

Der Kerberos-Client wird über die Konfigurationsdatei `/etc/krb5.conf` konfiguriert. In den Tests hatte die Datei folgenden Inhalt:

Listing 6.11: krb5.conf

```
[domain_realm]
  .linux.test = "LINUX.TEST"

[libdefaults]
  default_realm = "LINUX.TEST"

[realms]
  LINUX.TEST = {
    admin_server = "kerberos.linux.test"
    kdc = "kerberos.linux.test"
    master_kdc = "kerberos.linux.test"
  }
```

Nachdem mit dem Kommando `kinit Benutzername@linux.test` ein Kerberos Ticket vom Server geholt wurde, kann mit `mount -t nfs4 -o sec=krb5 nfs4.linux.test:/home /home` das exportierte Verzeichnis eingebunden werden.

6.6.3 Einrichten eines Kerberos Servers

Kerberos ist ein verteilter Authentifizierungsdienst der für unsichere Netze konzipiert wurde. Der Dienst Authentifiziert alle Kommunikationsbeteiligten und ist Plattformunabhängig. Das derzeitig aktuelle Kerberos-Protokoll Version 5 ist im RFC 4120 spezifiziert. Kerberos setzt ein funktionierendes Domain Name System und eine Zeitsynchronisation aller Kommunikationsbeteiligten vorraus. Ein Benutzerkonto im Kerberos hat folgendes Schema:

`Benutzername/Rolle@dns-domain`

Das Kommando `aptitude install bind9` installiert einen DNS Server unter Debian Linux. Die Domain während der Tests wird **linux.test** lauten. In der Datei **/etc/bind/db.linux.test** wird die Domain konfiguriert.

Listing 6.12: /etc/bind/db.linux.test

```
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns.linux.test. hostmaster.linux.test. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
        NS      ns
        MX      10      mail1.linux.test.
        MX      20      mail2.linux.test.
ns      IN      A       192.168.3.135
mail1   IN      A       192.168.3.131
mail2   IN      A       192.168.3.132
server  IN      A       192.168.3.135

kerberos        IN      A       192.168.3.135
afs             IN      A       192.168.3.135
_kerberos       in      TXT     "LINUX.TEST"
_kerberos-master._udp   IN      SRV     0 0 88 kerberos
_kerberos-adm._tcp      IN      SRV     0 0 749 kerberos
_kpasswd._udp   IN      SRV     0 0 464 kerberos
_kerberos._udp  IN      SRV     0 0 88 kerberos
LINUX.TEST.     IN      AFSDB 1 afs.linux.test.
```

Über die speziellen Kerberos-Einträge suchen die Kerberos-Anwendungen die KDC-Dienste. Auch die Rückwertsauflösung muss funktionieren. Dazu wird die Datei **/etc/bind/db.192.168.3** verwendet.

Listing 6.13: /etc/bind/db.192.168.3

```
1  ;
2  ; BIND reverse data file for local loopback interface
3  ;
4  $TTL    604800
5  @       IN      SOA     ns.linux.test. hostmaster.linux.test. (
6                                  1         ; Serial
7                             604800         ; Refresh
8                              86400         ; Retry
9                            2419200         ; Expire
10                            604800 )       ; Negative Cache TTL
11 ;
12 @       IN      NS      ns.linux.test.
13 1       IN      PTR     gw.linux.test.
14 135     IN      PTR     server.linux.test.
```

Der DNS Server ist jetzt fertig konfiguriert.

Mit dem Kommando `apt-get install krb5-kdc krb5-admin-server krb5-clients` werden die Kerberos Dienst installiert.

In der Kerberos-Welt wird der Administrationsbereich eines Server als **Realm** und ein Benutzerkonto als **principal** bezeichnet. Mit dem Kommandos `krb5_newrealm` und `kdb5_util create -s` wird ein neuer Administrationsbereich und mit `kadmin.local` und `addprinc test/admin` wird ein neues Benutzerkonto eingerichtet.

6.6.4 Einrichten eines Kerberos Clients unter Windows XP SP2

Der Kerberos Client kann unter `http://web.mit.edu/kerberos/dist/` heruntergeladen werden. Derzeitig wird der Client in der Version 3.2.2 angeboten. Die Installationsroutine des Kerberos Client fragt nach den Kerberos Einstellungen, die in Form einer Konfigurationsdatei `krb5.ini` von einer *URL* oder aus einer lokalen Datei geladen werden können.

Während des Tests hatte die Datei folgenden Inhalt:

Listing 6.14: krb5.ini

```
1  [domain_realm]
2    .linux.test  = "LINUX.TEST"
3
4  [ libdefaults ]
5    default_realm = "LINUX.TEST"
6
7  [realms]
8    LINUX.TEST = {
9      admin_server = "kerberos.linux.test"
10     kdc = "kerberos.linux.test"
11     master_kdc = "kerberos.linux.test"
12   }
```

Der Kerberos Client heißt im Windows **Network Identity Manager**. Mit diesem Programm können Kerberos Tickets vom Server geholt werden. Dazu muss der Benutzername, Passwort und der Default-Realm eingegeben werden.

7 Entscheidung

7.1 Ergebnis des Lasttests

Die Grafik wurde so skaliert, daß die Y-Achse die Bandbreite von 100 MBit/Sekunde darstellt. Dieser Wert ist die theoretisch maximal erreichbare Datenrate der Hardware des Testaufbaus.

$100 MBit/Sekunden = 12.5 MByte/Sekunde = 12800 KByte/Sekunde$

Der maximale Y Wert liegt bei 12800 KByte / Sekunde. Jeder Wert der darüber liegt, muss also aus einem Puffer gekommen sein.

7.1.1 CIFS/Samba

7.1.1.1 Linux Client

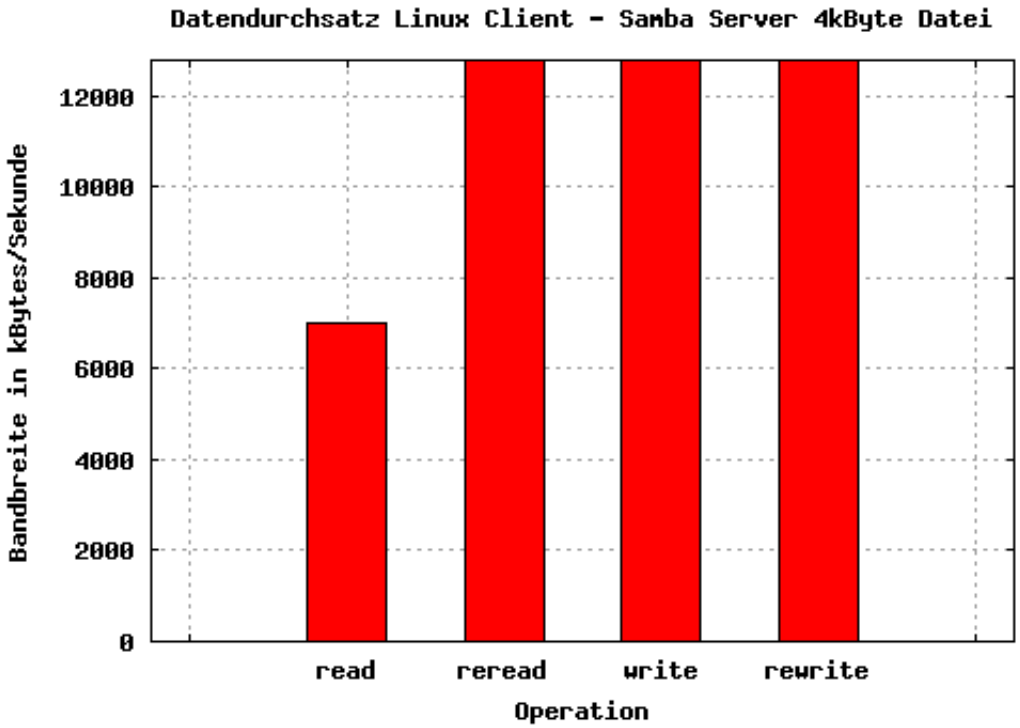

Abbildung 7.1: Linux Client - Samba Server 4kByte Datei

Der Einfluß des Puffers auf dem Client ist hier deutlich an den Werten zu erkennen. Eine Datei von 4KByte Größe passt vollständig in den Puffer, weshalb diese mit einer Datenrate von 540 MByte/Sekunde auch erstmal dort hinein geschrieben wurde.

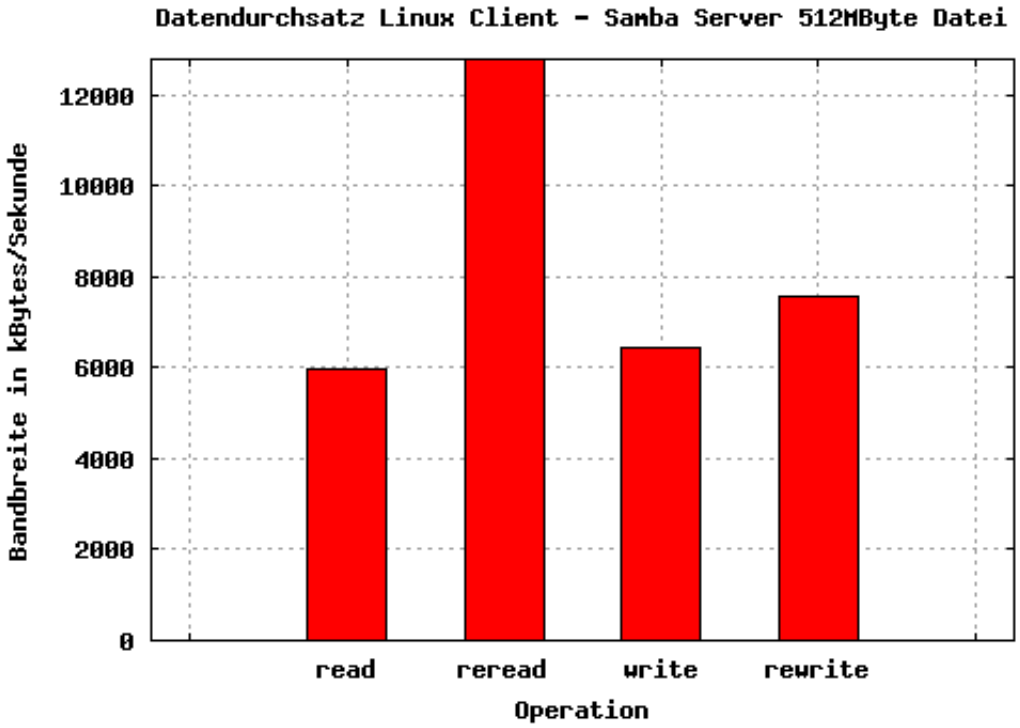

Abbildung 7.2: Linux Client - Samba Server 512MByte Datei

Mit der größeren Testdatei sind die Ergebnisse schon spürbar realistischer. Erstaunlich ist, daß die Werte deutlich unter den Erreichbaren liegen. Ich schließe darauf, daß die Implementierung von Samba nicht sehr effizient ist.

7.1.1.2 Windows Client

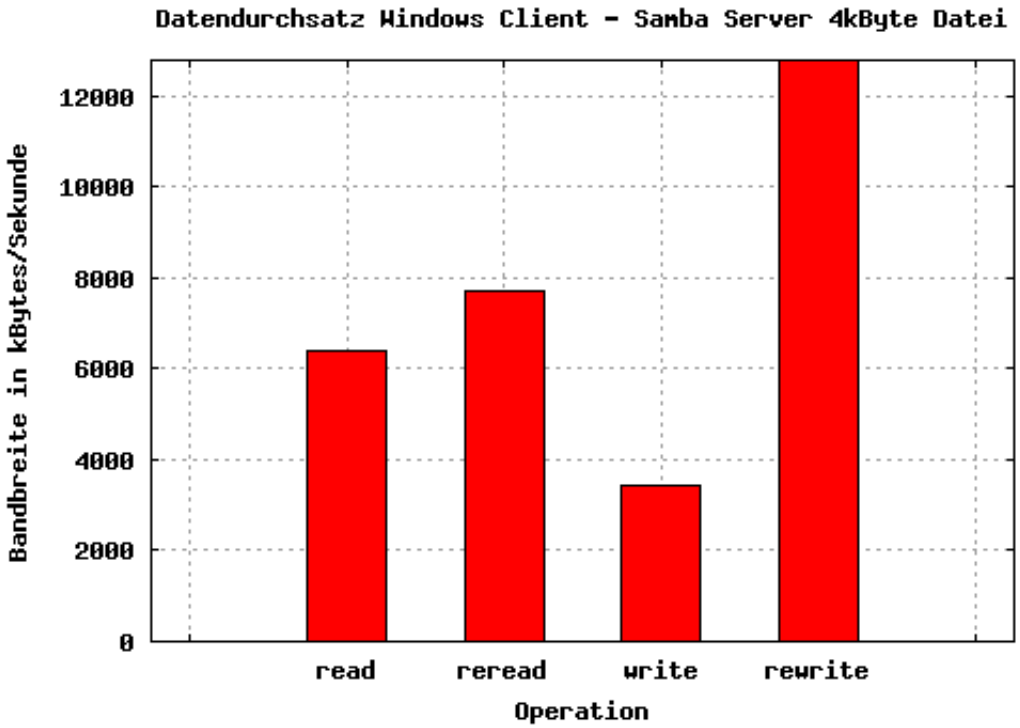

Abbildung 7.3: Windows Client - Samba Server 4kByte Datei

Das Testergebnis vom Windows Client legt die Vermutung nahe, daß die Windows Implementierung den Puffer nicht zum schreiben verwendet und damit wertvolle Performance verschenkt. Es ist sehr verwunderlich, weshalb die Entwickler nicht den Puffer nutzen.

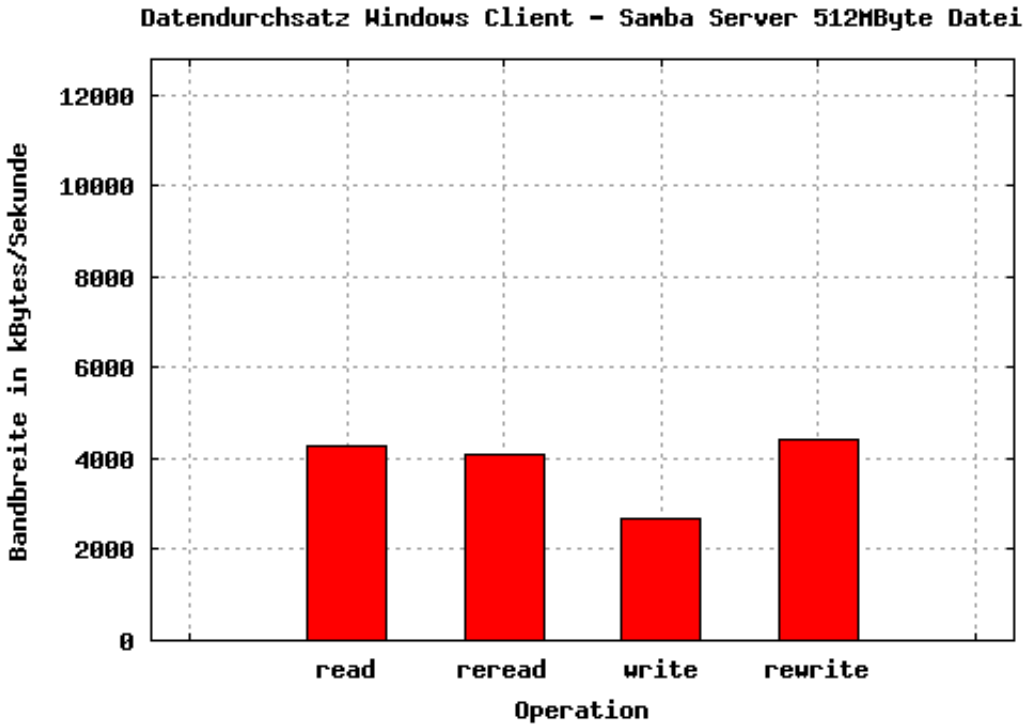

Abbildung 7.4: Windows Client - Samba Server 512MByte Datei

Samba hat sich in dem praktischen Test als langsam erwiesen. Datenraten von 2600 KByte / Sekunde um eine Datei zu schreiben liegt eindeutig hinter den Erwartungen.

7.1.2 NFSv3

7.1.2.1 Linux Client

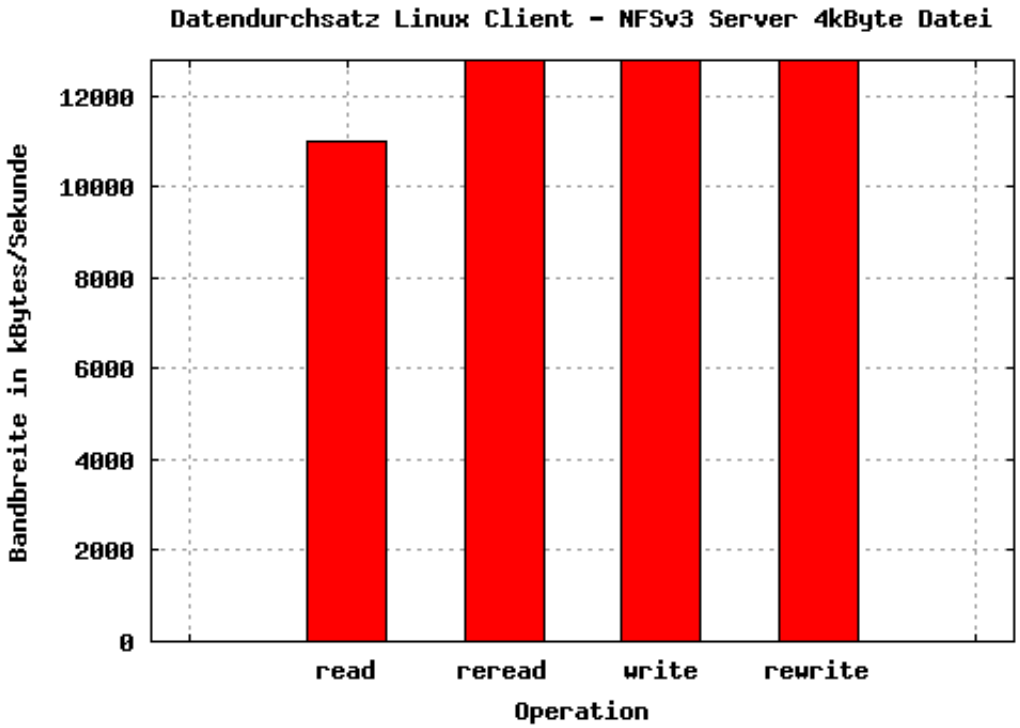

Abbildung 7.5: Linux Client - NFSv3 Server 4KByte Datei

Der Linux Client nutzt den Puffer um die Daten erst in diesen zu schreiben.

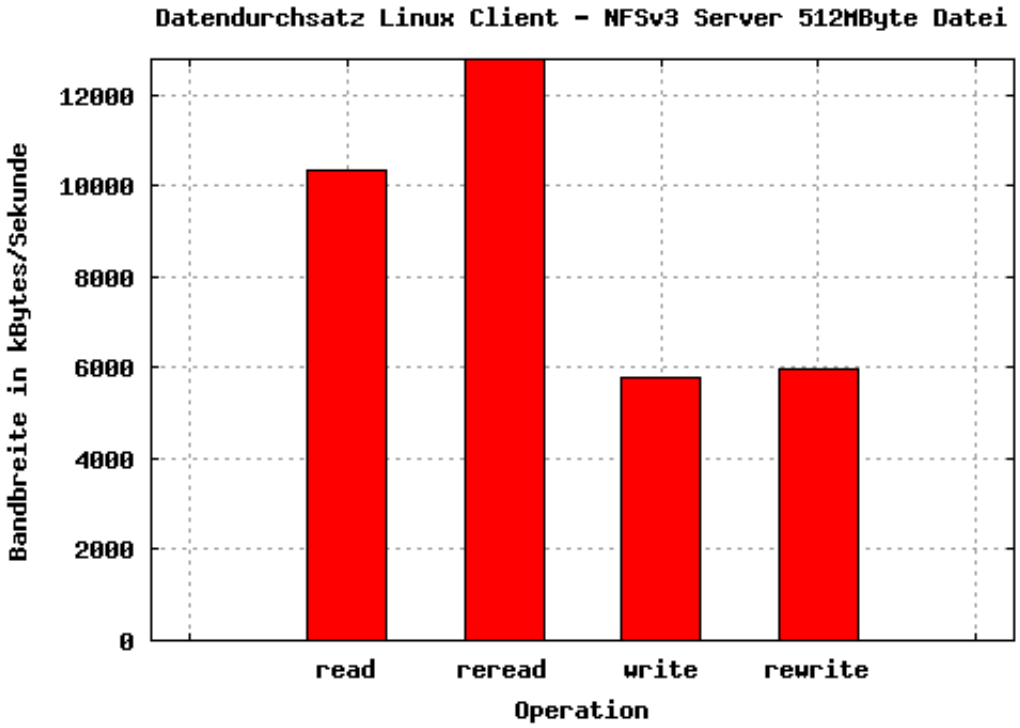

Abbildung 7.6: Linux Client - NFSv3 Server 512MByte Datei

NFSv3 unter Linux kann durch die hohe Netzwerkperformance überzeugen.

7.1.2.2 Windows Client

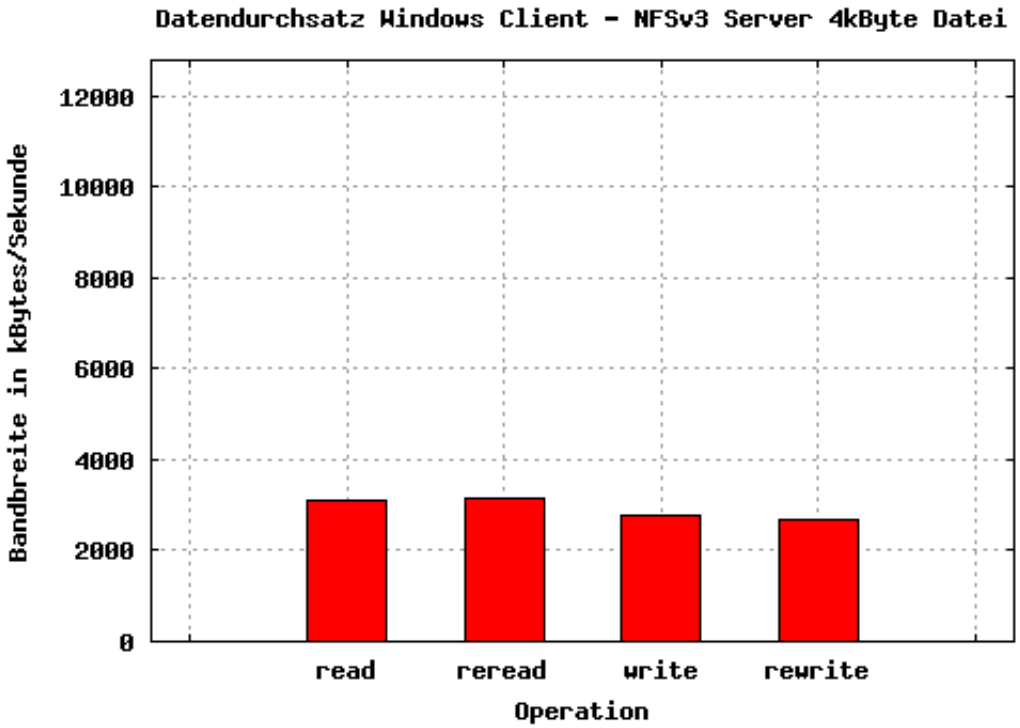

Abbildung 7.7: Windows Client - NFSv3 Server 4KByte Datei

Die Windows Implementierung von NFS nutzt keinen Puffer und ist dadurch sehr ineffizient. Datenraten von 2700 KByte / Sekunde sind in der heutigen Zeit inakzeptabel.

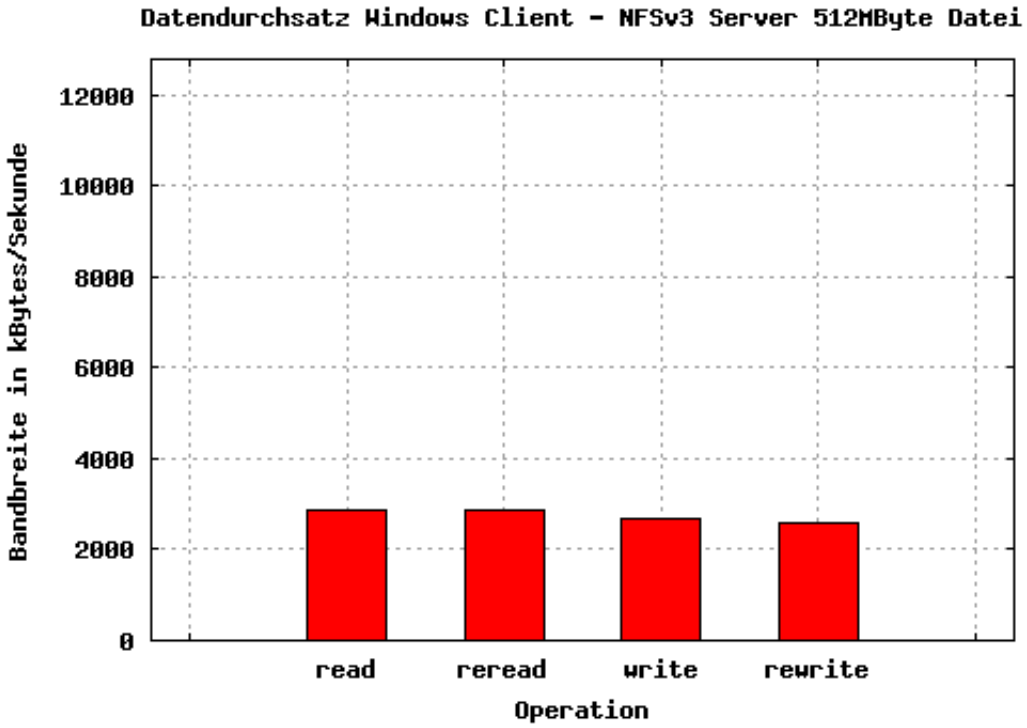

Abbildung 7.8: Windows Client - NFSv3 Server 512MByte Datei

Die Implementierung des NFS Client unter Windows enttäuscht durch ein sehr geringe Datenrate.

7.1.3 NFSv4

7.1.3.1 Linux Client

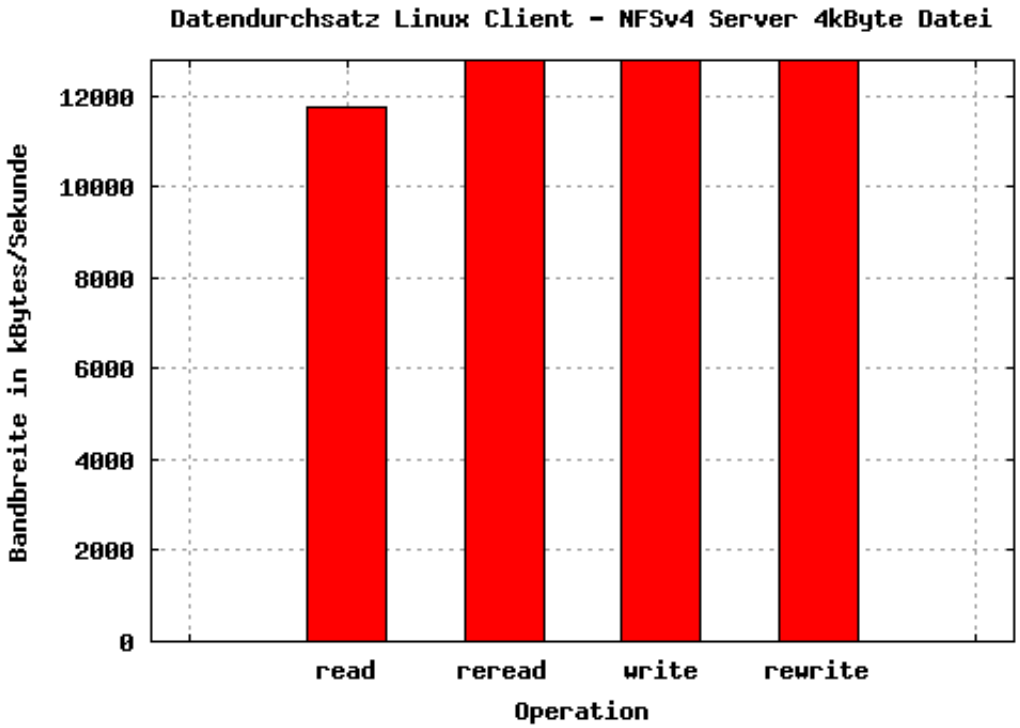

Abbildung 7.9: Linux Client - NFSv4 Server 4kByte Datei

Die Kombination eines Linux-Server und Client mit NFSv4 Netzwerkdateisystem war im gesamten Test die schnellste.

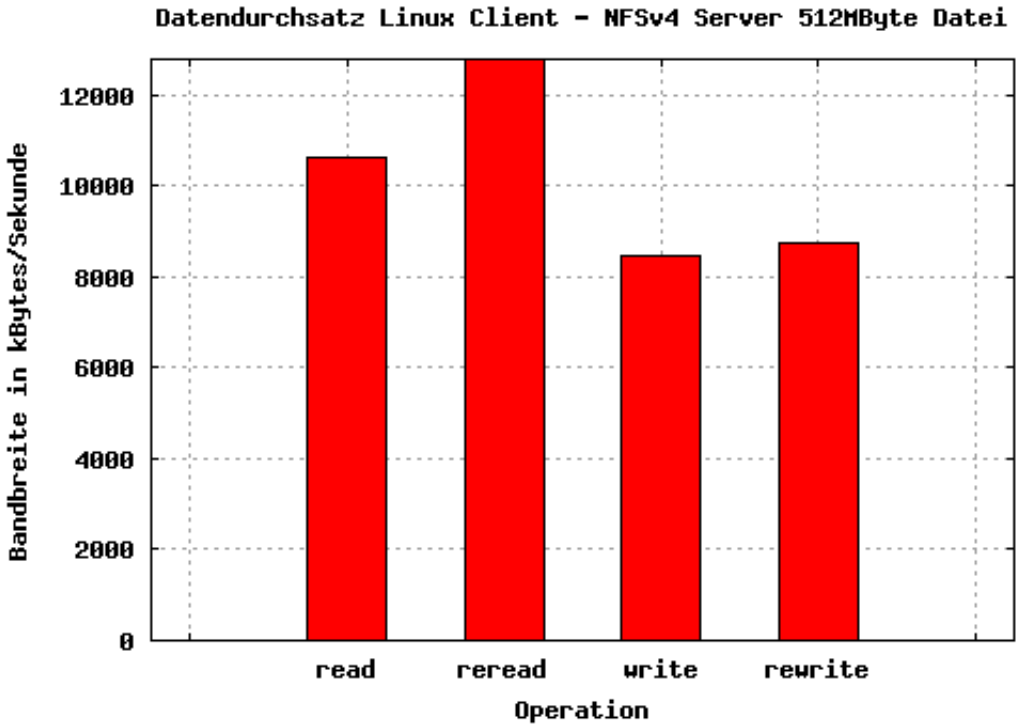

Abbildung 7.10: Linux Client - NFSv4 Server 512MByte Datei

Eine Rate von 10633 KByte/Sekunde bei einer 512MByte Datei stellt fast das Maximum von Ethernet dar und deutet darauf hin, daß NFSv4 ein sehr effizientes Netzwerkdateisystem ist.

7.1.4 OpenAFS

7.1.4.1 Linux Client

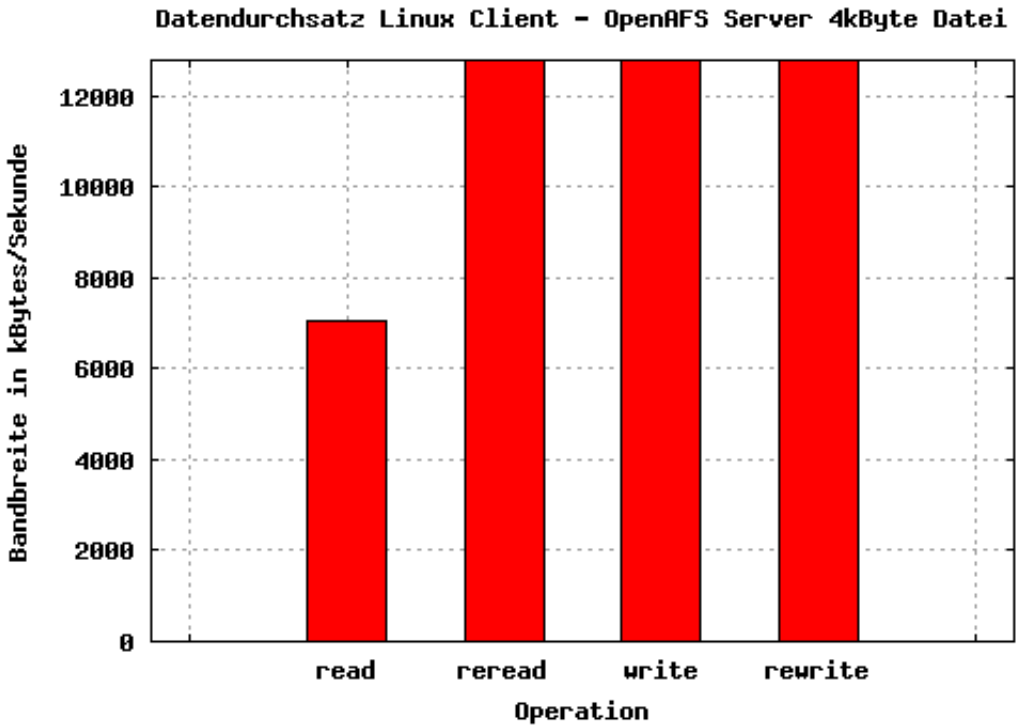

Abbildung 7.11: Linux Client - OpenAFS Server 4kByte Datei

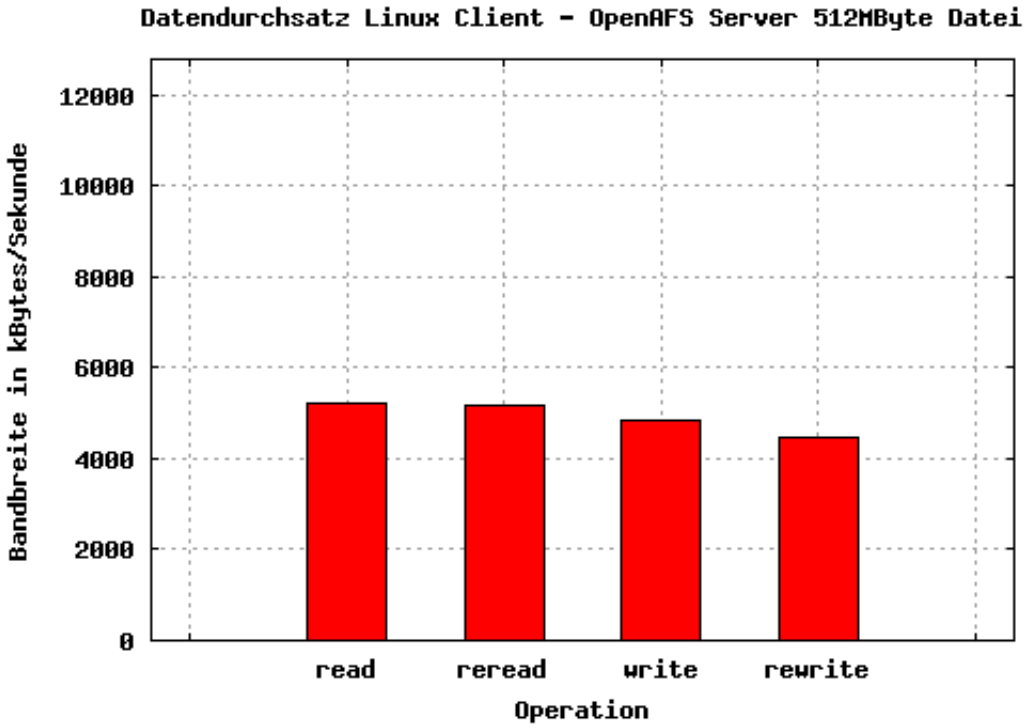

Abbildung 7.12: Linux Client - OpenAFS Server 512MByte Datei

An dem Testergebnis vom OpenAFS Server und Linux Client ist auffallend, daß die `wieder-lesen` Datenrate geringer ist, als die `lesen` Datenrate. Beim `schreiben` und `wieder-schreiben` sind ebenfalls seltsame Werte entstanden, die nur einen Schluß zulassen: Die Entwickler von OpenAFS nutzen den Puffer nicht optimal.

7.1.4.2 Windows Client

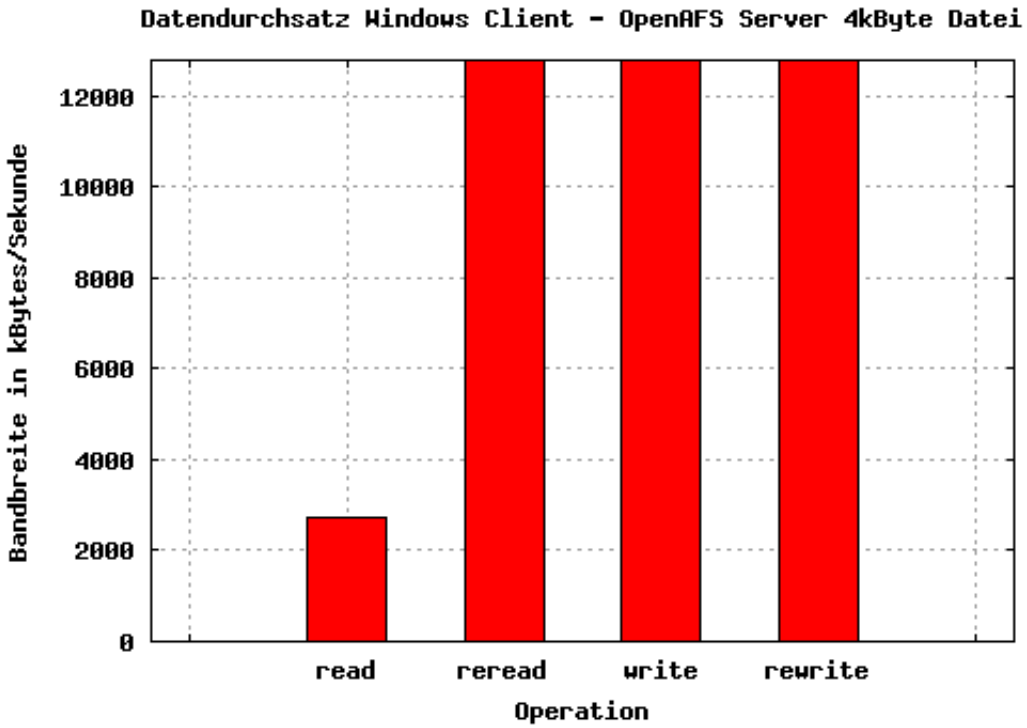

Abbildung 7.13: Windows Client - OpenAFS Server 4kByte Datei

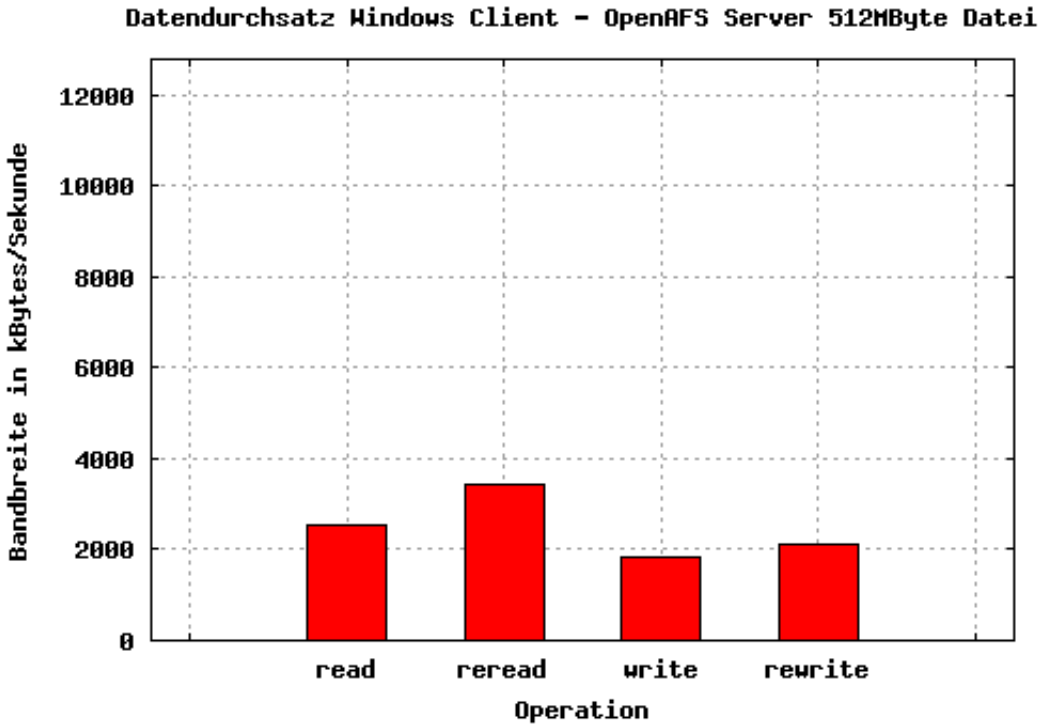

Abbildung 7.14: Windows Client - OpenAFS Server 512MByte Datei

Die Datenraten von OpenAFS unter Windows sind nicht sehr überzeugend und mit Werten um die 2000 bis 3000 KByte / Sekunde nicht akzeptabel.

7.1.5 Coda

7.1.5.1 Linux Client

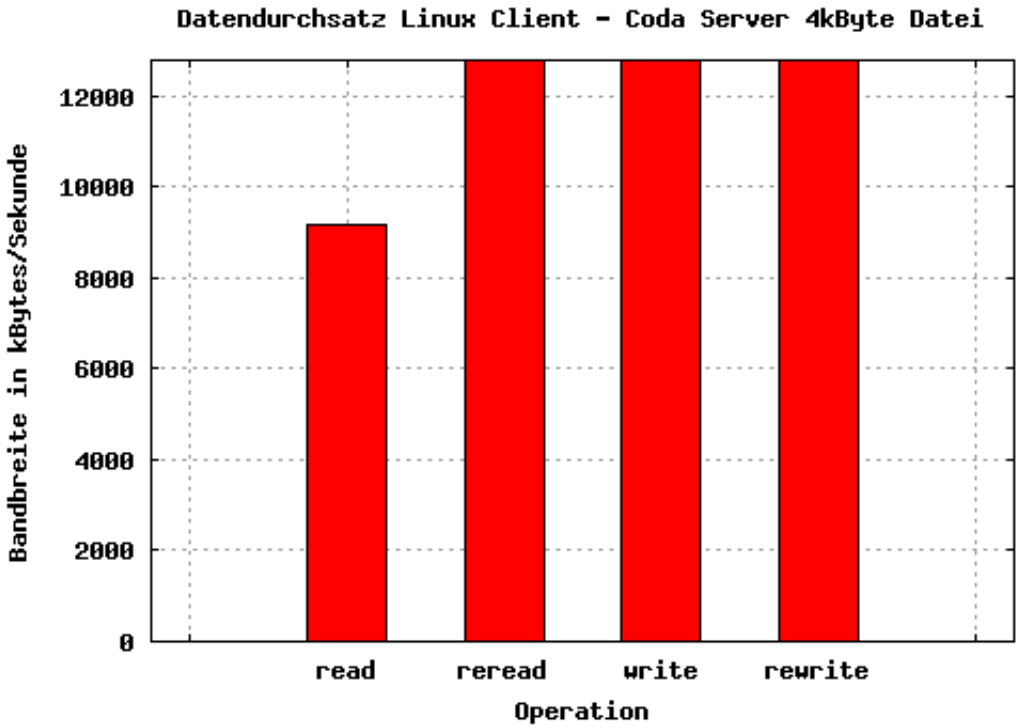

Abbildung 7.15: Linux Client - Coda Server 4kByte Datei

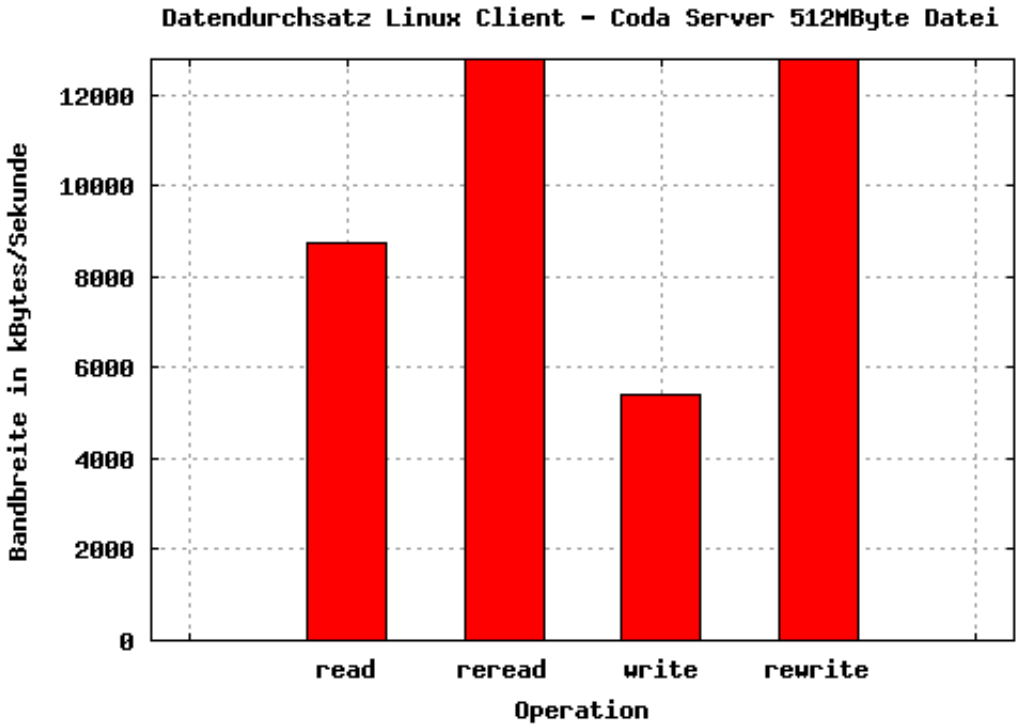

Abbildung 7.16: Linux Client - Coda Server 512MByte Datei

Die Linux Implementierung von Coda zeigt bei dem Test eine gute Datenrate. Als Nachfolger von OpenAFS ist an dem Testergebnis ersichtlich, das die Coda Entwickler die Datenrate verbessert haben.

7.2 Entscheidung für ein Netzwerkdateisystem

Die beste Datenrate zeigte ganz klar NFSv4. Auch der geringe Einrichtungs- und Verwaltungsaufwand spricht für den Nachfolger von NFSv3. Aber durch den fehlenden Windows-Client werden nicht alle Anforderungen erfüllt.

Somit ist CIFS/Samba mit Kerberos-Authentifizierung das Netzwerkdateisystem, welches die Anforderungen am besten erfüllt.

Durch den Einsatz von Kerberos werden zuverlässig *Man-in-the-Middle*-Attacken unterbunden und eine strenge Nutzerauthentifizierung eingehalten.

Mit dem Programm `autofs` kann der Homebereich nach einer erfolgreichen Anmeldung eingebunden werden.

CIFS/Samba kann seine Nutzerinformationen aus einem LDAP-Verzeichnis lesen und die Zugriffsrechteverwaltung basierend auf Posix-ACLs verwalten.

7.3 Implementierung an der HTWM Mittweida

Die Implementierung wird in einem virtuellen Host basierend auf der XEN Virtualisierungstechnologie realisiert.

Dazu wurde ein spezieller Linux Kernel installiert: `aptitude install xen-linux-system-2.6.26-2-xen-686 xen-utils xen-tools lvm2 python-xml`

Dieser Kernel bietet die nötige Unterstützung, um Gastbetriebssysteme parallel auf einem Rechner mit der Xen Virtualisierung zu betreiben.

Um einen Gast mit dem Betriebssystem Debian GNU/Linux in Xen zu installieren, wurde in der Datei `/etc/xen-tools/xen-tools.conf` folgende Einträge eingefügt:

Listing 7.1: xen-tools.conf

```
1 dir = /home/xen
2 size     = 10Gb      # Disk image size.
3 memory = 256Mb # Memory size
4 swap   = 256Mb   # Swap size
5 # noswap = 1      # Don't use swap at all for the new system.
6 fs       = ext3      # use the EXT3 filesystem for the disk image.
7 dist     = lenny      # Default distribution to install.
8 image  = sparse
```

Mithilfe des Kommandos `xen-create-image -hostname=samba` wird die virtuelle Festplatte in Form einer Datei erstellt und formatiert und eine Minimalinstallation des Betriebsystem Debian GNU/Linux Lenny wird über das Netzwerk durchgeführt.

Das Passwort für den Systembenutzer wurde auf `passwort` eingestellt.

Der Samba Server wurde aus der Paketverwaltung des Betriebssystems installiert und genau wie in dem praktischen Test konfiguriert.

Die virtuelle Maschine kann mit dem Kommando `xm create samba.cfg` gestartet werden.

Literaturverzeichnis

1001

1001: *Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods*. http://www.faqs.org/rfcs/rfc1001.html. – [Online; Stand 25. Januar 2010] 5.2.2

1002

1002: *Protocol standard for a NetBIOS service on a TCP/UDP*. http://www.faqs.org/rfcs/rfc1002.html. – [Online; Stand 25. Januar 2010] 5.2.2

BSI 2010a

BSI: *Falsches Exportieren von Dateisystemen unter Unix*. https://www.bsi.bund.de/cln_183/sid_8A45EC1B137859E82D1C3C6BC1047A6D/ContentBSI/grundschutz/kataloge/g/g03/g03010.html.
Version: 2010. – [Online; Stand 2. Maerz 2010] 5.4.2

BSI 2010b

BSI: *Sicherer Einsatz von SAMBA*. https://www.bsi.bund.de/ContentBSI/grundschutz/kataloge/m/m05/m05082.html.
Version: 2010. – [Online; Stand 25. Januar 2010] 5.2.2.1

Citi 2010

Citi: *NFSv4.1 Client for Windows*. http://www.citi.umich.edu/projects/nfsv4/windows/. Version: 2010. – [Online; Stand 25. Januar 2010] 5.5.3

iozone.org 2010

iozone.org: *IOZone Filesystem Benchmark*. http://iozone.org.
Version: 2010. – [Online; Stand 15. Februar 2010] 6.1

Kühnel 2004

Kühnel, Jens: *Samba 3 - Wanderer zwischen den Welten*. mitp-Verlag/Bonn, 2004 6.3.1

MSDN 2010

MSDN: *NT LAN Manager (NTLM) Authentication Protocol Specification*. http://msdn.microsoft.com/en-us/library/cc236621(PROT.10).aspx. Version: 2010. – [Online; Stand 10. Februar 2010] 5.2.2.1

NetAPP 2010

NetAPP: *Beschleunigter Zugriff auf gemeinsam genutzte Daten für Rechner-Cluster*. http://www.netapp.com/de/communities/tech-ontap/pnfs-1208-de.html. Version: 2010. – [Online; Stand 25. Januar 2010] 5.5.3

Nickolai Zeldovich

Nickolai Zeldovich, kolya@MIT.EDU: *Rx protocol specification draft*. http://web.mit.edu/kolya/afs/rx/rx-spec 5.1.2

openafs.org 2009

openafs.org: *OpenAFS Admin Guide*. http://docs.openafs.org/AdminGuide/index.html. Version: 2009. – [Online; Stand 18. November 2009] 5.1.3

P.A.Gloor 1989

P.A.Gloor: *Syncronisation in verteilten Systemen*. B.G.Teubner Stuttgart, 1989 5.4.2

Rechenberg 1999

Rechenberg, Peter: *Informatik Handbuch*. Hanser, 1999 3.2.2

Schneider 2005

Schneider, Thomas: *NFSv4 und Coda - eine Fallstudie verteilter Dateisysteme*. http://atmvs1.informatik.tu-muenchen.de/Members/spies/seminare/ss05-bszukunft/Ausarbeitung04.pdf. Version: 2005. – [Online; Stand 25. November 2009] 3.2.1

Smith 2001

Smith, Roderick: *Linux Samba Server Administration*. Sybex, 2001 5.2.2.2

Sun 2009

Sun, Microsystems: *Solaris ZFS Administration Guide*. http://docs.sun.com/app/docs/doc/817-2271/zfsover-1?a=view\&q=zfs. Version: 2009 3.1.2

Tannenbaum 1994

Tannenbaum, Andrew S.: *Moderne Betriebssysteme*. Carl Hanser Verlag München Wien, 1994 3, 3.1, 3.1.1

Wikipedia 2009

Wikipedia: *Liste von Dateisystemen — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Liste_von_Dateisystemen&oldid=66878829. Version: 2009. – [Online; Stand 25. November 2009] 4

Eidesstattliche Erklärung

Hiermit erkläre ich, daß ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Vetschau, den 19. Mai 2010

Ronny Gubatz

A Anhang

Listing A.1: Testergebnis

```
1
2  NFSv3 Server Windows Client
3  Blocksize       schreiben wied. Schreiben lesen    wied. Lesen
4  4096            2789      2684           3100     3161
5  524288          2696      2595           2894     2854
6
7  NFSv3 Server Linux Client
8  Blocksize       schreiben wied. Schreiben lesen    wied. Lesen
9  4096            546937    687814         11032    1320398
10 524288          5803      5959           10348    1671016
11
12 Samba Server Windows Client
13 Blocksize       schreiben wied. Schreiben lesen    wied. Lesen
14 4096            3431      142610         6397     7738
15 524288          2681      4429           4278     4082
16
17 Samba Server Linux Client
18 Blocksize       schreiben wied. Schreiben lesen    wied. Lesen
19 4096            539092    619019         6990     929050
20 524288          6442      7574           5978     1544412
21
22 OpenAFS Server Windows Client
23 Blocksize       schreiben wied. Schreiben lesen    wied. Lesen
24 4096            353730    372090         2740     335200
25 524288          1820      2102           2520     3420
26
27 OpenAFS Server Linux Client
```

Blocksize	schreiben	wied. Schreiben	lesen	wied. Lesen
4096	524733	584025	7038	962550
524288	4850	4474	5233	5189

Coda Server Linux Client

Blocksize	schreiben	wied. Schreiben	lesen	wied. Lesen
4096	532748	618332	9173	1258248
524288	5430	190284	8734	438332

NFSv4 Server Linux Client

Blocksize	schreiben	wied. Schreiben	lesen	wied. Lesen
4096	560376	593875	11754	1466089
524288	8475	8763	10633	1751612